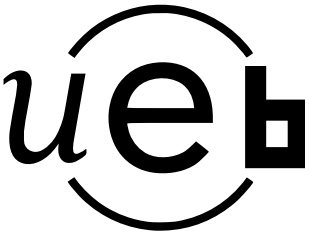
ANNÉE 2013



THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

École doctorale Matisse

présentée par

Nicolas MAILLET

préparée à l'unité de recherche INRIA/IRISA – UMR6074
Institut de Recherche en Informatique et Systeme Aleatoires

Comparaison *de novo* de données de séquençage issues de très grands échantillons métagénomiques.

Thèse soutenue à Rennes
le 19 décembre 2013

devant le jury composé de :

Philippe VANDENKOORNHUYSE
Professeur à l'Université de Rennes 1 / *Président du jury*
Ana Tereza Ribeiro de VASCONCELOS
Directrice de recherche au LNCC, Brésil / *Rapporteure*
Mihai POP
Professeur à l'Université du Maryland, États-Unis / *Rapporteur*
Olivier JAILLON
Chercheur CEA au Genoscope, Evry / *Examinateur*
Dominique LAVENIER
Directeur de recherche à INRIA, Rennes / *Directeur de thèse*
Pierre PETERLONGO
Chargé de recherche à INRIA, Rennes / *Co-directeur de thèse*

Nicolas MAILLET : *Comparaison de novo de données de séquençage issues de très grands échantillons métagénomiques,* Application sur le projet Tara Oceans, Décembre 2013

Ce travail est dédié à Patrick Ferreira.

RÉSUMÉ

La métagénomique vise à étudier le contenu génétique et génomique d'un échantillon provenant d'un environnement naturel. Cette discipline récente s'attache à étudier les génomes de différents organismes provenant d'un même milieu. La métagénomique pose de nouvelles questions, tant d'un point de vue biologique qu'informatique. Les masses de données générées par les études métagénomiques et la complexité des milieux étudiés nécessitent de développer de nouvelles structures de données et de nouveaux algorithmes dédiés.

Parmi les différentes approches existantes en métagénomique, la métagénomique comparative consiste à comparer plusieurs métagénomes afin d'en connaitre les divers degrés de similarité. Lorsque cette comparaison se base uniquement sur le contenu brut des échantillons, sans faire appel à des connaissances externes, on parle de métagénomique comparative *de novo*.

L'objectif des travaux que nous proposons est de développer une méthode permettant d'extraire les séquences similaires entre deux jeux de données métagénomiques, où chaque jeu peut être composé de centaines de millions de courtes séquences d'ADN. La comparaison proposée consiste à identifier les séquences d'un premier jeu similaires à au moins une séquence d'un second jeu.

Afin d'être rapide et économe en mémoire, l'implémentation de notre méthode a nécessité la conception d'une nouvelle structure d'indexation, basée sur le filtre de Bloom. Le logiciel final, nommé Compareads, a une consommation mémoire faible (de l'ordre de quelques Go) et peut calculer l'intersection de deux échantillons de 100 millions de séquences chacun en une dizaine d'heures. Notre méthode est une heuristique qui génère un faible taux de faux positifs.

Le logiciel Compareads est dédié à l'analyse de grands jeux de données métagénomiques. À l'heure actuelle, il est le seul outil capable de comparer de tels jeux. Compareads a été appliqué sur plusieurs projets métagénomiques. Notre outil produit des résultats robustes, biologiquement exploitables et en accord avec diverses méthodes fondamentalement différentes. Il est actuellement utilisé de manière intensive sur les échantillons provenant de l'expédition Tara Oceans. Sur ce projet, notre méthode a permis de mettre en évidence que les grands systèmes océaniques influent sur la répartition globale des microorganismes marins.

Bioinformaticists should focus on exploring
new ways to interpret metagenomic sequence data,
with continuous reference to the needs of the community.

— Narayan Desai *et al.* 2012 [1]

REMERCIEMENTS

Il est d'usage de commencer les remerciements par ses directeurs de thèse. Mais, si je tiens à les remercier avant toute autre personne, ce n'est pas par tradition, mais bien pour tout ce qu'ils m'ont apporté durant ces trois années. Que ce soit d'un point de vue professionnel, scientifique ou humain, Dominique et Pierre ont été motivants, formateurs et source d'inspiration ; pour tout cela, je les remercie.

Mes remerciements vont ensuite vers l'ensemble de l'équipe. Initialement symbiose puis scindée en Dyliss et Genscale, les membres de cette équipe furent toujours parfaits. Ne changez rien ! Au sein de symbiose, je remercie particulièrement Claire Lemaitre, Clovis Galiez et Vincent Picard pour toute l'aide apportée sur les aspects statistiques et probabilistes de mes recherches. Je tiens aussi à remercier Guillaume Collet pour ses nombreux conseils, surtout en terme de communication scientifique, et Rayan Chikhi et Guillaume Rizk pour l'aide apportée sur la conception et l'analyse de la structure de données développée durant mon doctorat.

Mes remerciements vont aussi à Eric Pelletier, pour son aide lors de la rédaction du manuscrit, et Thomas Vannier, pour tout le travail réalisé sur les tests et l'utilisation de notre méthode.

J'adresse aussi un grand merci à Damien Eveillard pour toutes les excellentes discussions que nous avons eues ! Je souhaite ensuite remercier Fred pour la création du symbole $\widetilde{\cap}$ et Ludo, Adrien, Olivier et Stéphane, pour avoir toujours été là quand j'ai eu besoin de décompresser ! Je remercie aussi chaleureusement mes deux collègues de bureau, Sylvain Prigent et Guillaume Chapuis, pour tous les moments passés ensemble, qu'ils aient été bons ou excellents ! Dans la même veine, merci à Coraline Lafon, Valentin Wucher et Paulin Fournier, pour les nombreuses discussions agrémentées de caféine et de nicotine.

Merci aussi à tous mes amis, dont beaucoup sont déjà mentionnés plus haut. Merci donc à Barbara, Camille, Amélie, Richard, Laurent, Jo, Ludi, Tibo, Romain, Polo, Julie, Mélanie, Colin, Dups, ma GIN Sprint, Charles, Élodie, Cécile, Pierre, Mathieu, Joseph, Aurore, Raluca, Morgane, Maud, Damien et à tous ceux que j'oublie momentanément :)

Mes remerciements ne seraient pas complets sans un mot pour ma famille. Je remercie donc infiniment mes parents, pour avoir su m'écouter, même sans toujours comprendre, quand j'avais besoin d'expliquer mon travail, et mon frère, keep it up bro' !

Merci à vous tous, sans qui ces trois années n'auraient pas pu se dérouler dans d'aussi bonnes conditions !

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ALGORITHMES

1 INTRODUCTION

Étudier le vivant, comprendre les interactions entre différents organismes, analyser les phénomènes comme l'évolution des espèces, permet de mieux comprendre le monde dans lequel nous évoluons et conduit à des améliorations au niveau de la gestion de l'environnement, de la médecine ou encore de l'agro-alimentaire. La vie se présente sous une infinité de formes et la biologie s'intéresse à des processus allant du niveau microscopique au niveau macroscopique, de la molécule à l'écosystème, en passant par la cellule, l'organisme et la population.

Pour cela, la biologie tend à créer toujours plus de données. Déjà durant le XIX^{e} siècle, les expéditions du H.M.S Beagle [2], de 1826 à 1843, et du H.M.S Challenger [3], de 1872 à 1876, ont chacune nécessité des dizaines d'années d'étude, entre autre car la quantité de données à analyser était très importante. Ces deux grandes expéditions océanographiques sont restées célèbres : la première, avec le H.M.S Beagle, est connue pour avoir embarqué à son bord Charles Darwin, qui y a établit les bases de sa théorie de l'évolution ; la seconde, avec le H.M.S Challenger, est souvent citée comme la première expédition d'océanographie moderne [4]. Cette expédition a en effet allié des analyses physiques, chimiques et biologiques.

Ces dernières décennies, l'informatique a gagné une place prépondérante au sein de nombreuses disciplines dont la biologie. Un nouveau champ disciplinaire est apparu : la bioinformatique. Cette discipline a pour but d'utiliser l'outil informatique pour résoudre des problèmes biologiques. L'augmentation de la puissance informatique et la création d'algorithmes spécialisés pour la biologie ont conduit à de nombreuses avancées, par exemple au niveau génomique. Mais, avec ces nouvelles possibilités, de nouvelles questions apparaissent et de plus en plus de données peuvent être générées. Les analyses *in silico*, par analogie avec *in vitro* ou *in vivo*, conduisent à la fois à une masse grandissante de données mais aussi à un traitement beaucoup plus rapide et systématique de ces données.

Le séquençage d'un génome, qui permet d'obtenir numériquement l'ADN d'un individu, est de plus en plus rapide, et de moins en moins coûteux à réaliser. Avec le développement des techniques d'analyses des génomes, l'idée d'analyser non plus le génome d'une espèce, mais l'ensemble des génomes provenant d'un même milieu, a émergé. Certains milieux sont composés d'un très grand nombre d'organismes microscopiques. Par exemple, dans un litre d'eau de mer, on estime qu'il y a plusieurs milliards de bactéries. La très grande majorité des espèces microscopiques nous est encore inconnue et est très difficilement cultivable en laboratoire [5]. Pouvoir analyser en une seule fois l'ensemble des génomes évoluant dans un milieu est donc particulièrement motivant ; c'est ainsi qu'est apparue la métagénomique. Dans un projet métagénomique, un échantillon biologique contient différentes espèces, et une partie des génomes contenus dans le milieu étudié est numérisée. D'un point de vue informatique, un échantillon métagénomique consiste alors en un très grand nombre, par exemple plusieurs millions, de courtes séquences d'ADN, c'est à dire de mots de quelques centaines de lettres pouvant chacune être A, C, G ou T.

Pour comparer deux génomes, on recherche généralement des similarités au niveau de l'ADN : plus les séquences de deux génomes sont similaires, plus ils sont considérés comme proches. En métagénomique, chaque échantillon peut comporter un très grand nombre de gé-

nomes. Intuitivement, on peut penser que deux échantillons métagénomiques provenant d'un même milieu ont une composition en espèce très proche. D'un point de vue informatique, on s'attend alors à ce que de nombreuses séquences entre les deux jeux de données soient similaires. À l'inverse, deux échantillons provenant de milieux très différents comportent sans doute moins d'espèces communes et les séquences de ces échantillons sont donc moins similaires. La question qui découle de cette intuition est alors : comment comparer deux échantillons métagénomiques ?

Les travaux présentés dans cette thèse portent sur la comparaison d'un très grand nombre de séquences d'ADN. Plus précisément, la méthode développée cherche à comparer deux échantillons métagénomiques sur la base des séquences proches ou identiques que ces deux ensembles partagent. Si ce calcul semble simple sur des petits jeux de données, le but est ici de pouvoir comparer, en un temps raisonnable, des jeux de données composés de centaines de millions de séquences chacun.

1.1 HISTORIQUE ET GÉNÉTIQUE

C'est en 1868 que le biologiste suisse Friedrich Miescher décrit pour la première fois les acides nucléiques [6]. Il baptise "nucléine" cette substance riche en phosphore et aux propriétés acides. En 1889, l'Allemand Richard Altmann arrive à extraire des protéines de cette nucléine ainsi qu'une substance acide : l'acide nucléique [7]. En 1896, l'Allemand Albrecht Kossel découvre dans l'acide nucléique les quatre bases azotées [8] : l'adénine (notée A), la thymine (notée T), la cytosine (notée C) et la guanine (notée G).

Pendant une cinquantaine d'années, la structure des acides nucléiques a été fortement étudiée [9], notamment grâce aux travaux de Levene [10], Jones [11], Feulgen [12], Astbury [13]... La structure et le rôle fondamental des acides nucléiques dans le vivant ont été saisis en 1953 grâce aux observations de Franklin [14, 15], Wilkins [16] et Watson & Crick [17, 18].

On sait désormais que le noyau des cellules comporte une substance riche en phosphore et en acide nucléique : l'ADN. Cette substance a une structure en double hélice et certaines parties, les gènes, sont transcrites en une autre molécule nucléotidique : l'ARN. Holley, Khorana et Nirenberg découvrent que chaque triplet de nucléotides de l'ARN code pour un acide animé : c'est ce qu'on nomme le code génétique. Ces triplets de nucléotides sont appelés codons. Par exemple, le codon GGA code pour l'acide aminé Glycine. La succession de ces acides aminés forme une protéine qui aura un rôle précis dans l'organisme. Les gènes codent ainsi pour des protéines et l'ADN sert de support à l'information génétique. En 1968, Holley, Khorana et Nirenberg reçoivent le prix Nobel de physiologie ou médecine pour avoir déchiffré le code génétique et compris ses fonctions dans la synthèse protéique.

Avant la découverte du rôle de l'ADN, la génétique était une science de l'hérédité, basée sur les ressemblances géniteur-descendance. Après cette découverte, la génétique est entrée dans une nouvelle ère consistant à étudier les gènes et leur évolution : la génétique moléculaire. Les séquences d'ADN évoluent au cours du temps, et les gènes que portent ces séquences subissent des mutations. La génétique étudie entre autres ces variations et leurs effets sur les organismes concernés.

Cette nouvelle génétique repose sur l'ordre d'enchaînement des nucléotides d'une séquence d'ADN ; c'est ce qu'on appelle le séquençage.

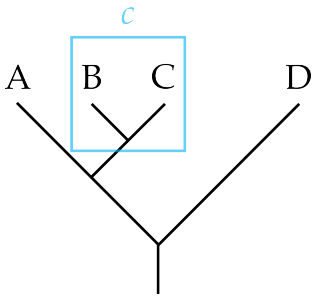


FIGURE 1: Représentation de l'arbre phylogénétique des quatre espèces A, B, C et D et d'un clade *c*. Dans cet arbre, les espèces B et C sont les plus proches d'un point de vue évolutif et l'espèce D est la plus éloignée des autres. Le clade *c* regroupe les deux espèces B et C.

1.2 RÉVOLUTION GÉNOMIQUE

Avec la découverte des premières maladies ayant une origine génétique, comme la trisomie 21 en 1959 [19], la génétique moléculaire gagne en popularité. En 1977, deux techniques de séquençage rapide sont proposées indépendamment. La méthode de l'équipe de Maxam et Gilbert [20] consiste à découper chimiquement l'ADN pour le lire, nucléotide par nucléotide. La méthode de Sanger *et al*. [21] consiste à rajouter des nucléotides spéciaux lors de la réplication de l'ADN, processus qui se déroule au sein de toute cellule et permet de recopier à l'identique une molécule d'ADN. Ces deux techniques sont connues sous le nom de première génération de séquenceurs.

1.2.1 De la génétique à la génomique

En génétique, on considère que l'ADN contient toute l'information suffisante pour permettre aux organismes de vivre et de se reproduire. De plus, on considère que deux séquences très proches ont de grandes chances d'avoir un rôle similaire. Le séquençage permet alors de comparer des gènes provenant d'espèces différentes. L'étude des séquences d'ADN devient un outil de classification des espèces à travers la phylogénétique. La phylogénétique est une branche de la phylogénie, qui consiste à comprendre et représenter l'histoire évolutive d'espèces. La phylogénie était entre autres basée sur les observations directes de spécimens. La phylogénétique a pour objectif de rendre compte des proximités entre espèces d'un point de vue génétique.

Une étude phylogénétique conduit à l'obtention d'un arbre phylogénétique, représentatif des liens de parenté entre les espèces étudiées (voir figure 1). Dans un tel arbre, un nœud représente l'ancêtre commun de ses descendants. Le nombre de nœuds entre deux branches indique alors le degré de parenté entre les espèces portées par ces branches : plus il y a de nœuds entre deux espèces, plus l'ancêtre commun à ces espèces est ancien et plus les espèces sont éloignées.

Dans un tel arbre, un clade est un regroupement d'espèces partageant un ancêtre commun. Les clades ont tendance à remplacer peu à peu la notion de taxon. Un taxon est défini comme un groupe conceptuel regroupant tous les organismes possédant des caractères en commun. La plupart du temps, ces caractères sont morphologiques. Il arrive que des espèces proches morphologiquement, donc d'un même taxon, soient finalement très éloignées génétiquement et donc appartiennent à des clades différents.

On peut représenter différemment une phylogénie. Le dendrogramme figure 2 reprend les mêmes espèces que celles de l'arbre figure 1. Ce dendrogramme informe sur la distance évolutive entre les espèces : plus les branches sont longues, plus les distances sont importantes.

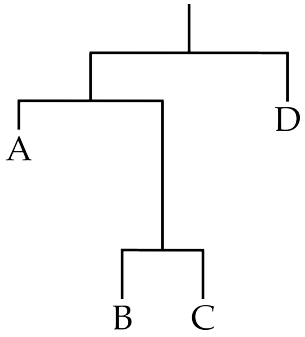


FIGURE 2: Dendrogramme équivalent à l'arbre phylogénétique figure 1 pour les quatre espèces A, B, C et D.
Dans ce dendrogramme, les espèces B et C sont les plus proches d'un point de vue évolutif et l'espèce D est la plus éloignée. La taille des branches représente la distance évolutive ; on peut voir que l'espèce B et C sont très éloignées de A.

Pour réaliser ce genre d'étude au niveau génétique, il faut comparer des séquences d'ADN entre elles. Or, comparer deux séquences d'ADN n'est pas une chose aisée. Un des moyens est d'aligner une séquence sur une autre de manière à minimiser le nombre de changements nécessaires pour passer d'une séquence à l'autre. Les changements peuvent être des substitutions, un nucléotide est remplacé par un autre ; des insertions, *i.e.* un nucléotide est ajouté dans la séquence ; ou, à l'inverse, des délétions, *i.e.* un nucléotide disparait.

Les insertions et délétions influent potentiellement plus sur la séquence protéique, une fois traduite, que les substitutions. Pour rappel, les nucléotides sont lus par triplets. Or, ajouter ou enlever un nucléotide décale totalement le reste de la séquence. Par exemple, AACGCG est lue AAC (qui code pour l'Asparagine) puis GCG (qui code pour l'Alanine). La protéine qui en découle est donc composée d'une Asparagine et d'une Alanine. Si une insertion arrive et que la séquence devient ACACGCG, la séquence lue sera alors ACA puis CGC, qui codent respectivement pour une Thréonine et une Arginine. La séquence protéique finale est ainsi différente et n'assurera probablement plus sa fonction initiale. Ce phénomène est appelé "décalage du cadre de lecture" et entraîne souvent la mort de la cellule portant la mutation : la mutation ne se transmet donc pas. Les substitutions ne changent pas le cadre de lecture et sont donc plus souvent viables pour la cellule.

Quand on aligne deux séquences d'ADN, on cherche à minimiser le nombre de mutations entre les séquences et particulièrement les insertions ou délétions. On peut aligner les séquences sur toute leur longueur, on parle alors d'alignement global. On peut aussi chercher quelles sous-parties des séquences s'alignent le mieux entre elles, on parle alors d'alignement local. La figure 3 représente ces différents types d'alignements. En 1990, Altschul *et al*. ont proposé un logiciel d'alignement local nommé BLAST [22] : ce logiciel est devenu le plus connus en bio-informatique.

Les alignements peuvent se faire sur plus de deux séquences, on parle alors d'alignements multiples. À partir d'un alignement multiple, on peut déduire une séquence consensus. Cette séquence est le calcul, pour chaque position, du nucléotide apparaissant le plus souvent. Dans un arbre phylogénétique, cette séquence consensus peut servir à représenter un ancêtre commun.

À la fin des années 90, les technologies de séquençage ont évolué et permettent désormais de séquencer des génomes entiers, par exemple *Saccharomyces cerevisiae* en 1996 [23], *Escherichia coli* en 1997 [24] et *Arabidopsis thaliana* en 2000 [25]. Dès lors, une nouvelle discipline apparaît, consistant non plus à étudier des gènes d'une ou plusieurs espèces, mais a étudier

```
AAACGCTTGCTGA
TACCTAGCTGATT
```

(a) Les séquences ne sont pas alignées.

```
AAACGCTTGCTGA--
.||  ||.|||||
-TAC-CTAGCTGATT
```

(b) Les séquences sont alignées globalement.

```
···CTTGCTGA···
  ||.|||||
···CTAGCTGA···
```

(c) Les séquences sont alignées localement.

FIGURE 3: Représentation d'alignements de séquences. En 3a, les séquences ne sont pas alignées. L'alignement global (3b) a fait intervenir une délétion (représentée par espace vide) et deux substitutions (représentées par le symbole ·). L'alignement local (3c) ne nécessite qu'une substitution mais ne comporte pas les séquences en entier.

l'ensemble des gènes, *i.e.* le génome, d'un organisme : on parle alors de génomique. La génomique à ouvert la porte à plusieurs autres disciplines. Par exemple, en étudiant l'ensemble des molécules qu'un organisme peut produire, suivant son ADN, on peut reconstruire un réseau représentant comment ces molécules interagissent les unes avec les autres. Ce type de réseau, nommé réseau métabolique, a bénéficié des progrès de la génomique [26].

1.2.2 Séquençage haut débit de génomes

L'arrivée des séquenceurs de nouvelle génération a accéléré la création de données génomiques. Ces technologies apparues entre 2005 et 2010 sont dites de haut débit, c'est à dire qu'elles produisent des millions de séquences en quelques dizaines d'heures, pour un coût bien moins élevé qu'auparavant. Les séquenceurs de nouvelle génération sont souvent abrégées en HTS, pour *High-Throughput Sequencing*, ou NGS, pour *Next-Generation Sequencing*. Ces approches sont massivement parallèles et peuvent séquencer des centaines de milliers de fragments d'ADN simultanément. Les deux techniques prépondérantes en NGS sont Roche 454 et Illumina.

Les séquenceurs 454 sont basés sur le pyroséquençage. Cette technique consiste à utiliser un seul brin de la double hélice d'ADN à séquencer puis à mimer le mécanisme de réplication de l'ADN. Ce mécanisme cellulaire sert à copier à l'identique une molécule d'ADN. On place l'ADN simple brin à séquencer en présence des protéines utilisées dans les cellules pour répliquer l'ADN mais sans nucléotide : la réplication ne se fait donc pas. On ajoute alors un seul type de nucléotide, par exemple A. Si un A est attendu sur le brin à dupliquer, il va être incorporé. La réaction d'incorporation produit une réaction lumineuse. Si on ne capte aucune lumière, c'est que le A n'était pas le nucléotide attendu. On nettoie l'ensemble, et on recommence avec un autre nucléotide. En 454, cette réaction se déroule simultanément sur de nombreuses molécules d'ADN. En 24 heures, les séquenceurs 454 produisent ainsi quelques millions de séquences de plusieurs centaines de nucléotides (ou paires de bases, abrégé en pb), pour un coût de quelques dollars par million de nucléotides.

Les séquenceurs Illumina fonctionnent différemment. Plutôt que d'ajouter les nucléotides type par type, on ajoute ici les quatre nucléotides simultanément, mais ces nucléotides sont couplés à une molécule fluorescente par type. Un seul des quatre nucléotides s'incorpore, et on nettoie alors l'ensemble. La couleur émise détermine le nucléotide inséré, et on recommence la procédure pour le nucléotide suivant. Cette technique produit plusieurs centaines de millions voire un milliard de courtes séquences (50 à 200 pb) en quelques jours, pour un coût de 5 à 15 centimes de dollar par million de paires de bases.

D'autre techniques de NGS, dont Ion Torrent et SOLiD offrent différents ratios en terme de nombre de séquences, de taille de séquences ou de prix par paire de bases. Cependant, à l'heure actuelle, 454 et Illumina restent les plus utilisées. Les NGS ont abaissé le coût de séquençage. En 2001, le séquençage d'un génome de taille similaire au génome humain coûtait autour de 100 millions de dollars. Aujourd'hui séquencer le même génome coûte moins de

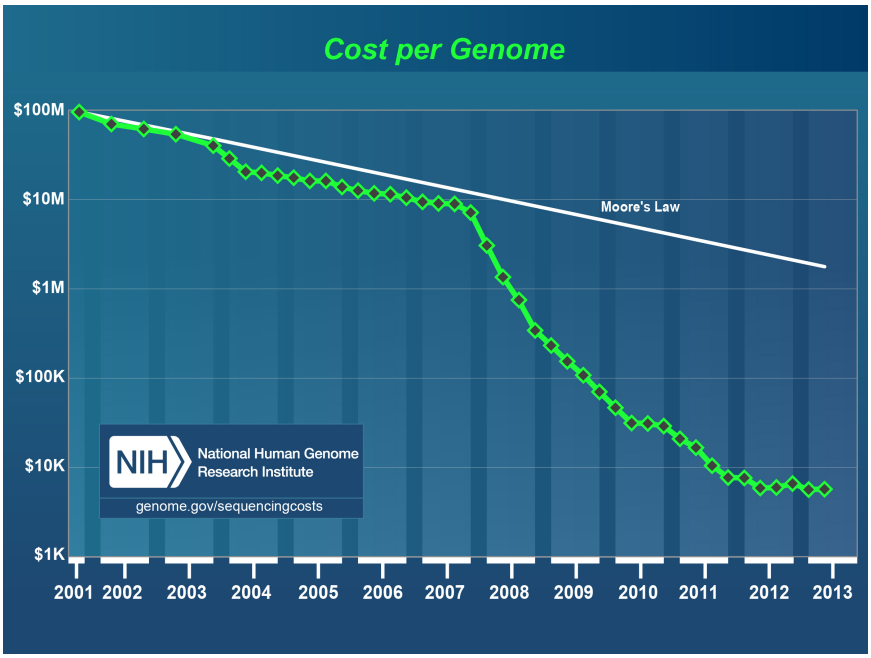


FIGURE 4: Représentation du coût de séquençage d'un génome de taille similaire au génome humain (échelle logarithmique). Depuis 2001, le prix pour séquencer un génome n'a cessé de décroître. Entre 2007 et 2008, la diminution est accélérée, conduisant à la création de plus en plus de données. La loi de MOORE symbolise l'évolution parallèle de la puissance de calcul disponible pour traiter ces données. On voit qu'après 2008, l'évolution des ressources informatiques ne suit plus la diminution des coûts de production génomique : l'évolution de la puissance brute informatique n'est plus suffisante pour traiter l'ensemble des données générées ; de nouveaux algorithmes doivent être mis au point.

(Source : www.genome.gov/sequencingcosts)

10 000 dollars [27] (voir figure 4).

Ces évolutions technologiques créent toujours plus de données. On parle parfois de "datanami", compression de tsunami et de *data* (données en anglais) pour exprimer l'énorme quantité de données biologiques ainsi générées. Cette quantité pose des défis à l'informatique. Il est en effet nécessaire de concevoir de nouvelles structures de données et des algorithmes toujours plus efficaces pour traiter les informations produites par les séquenceurs de nouvelle génération. En effet, l'évolution de la puissance de calcul (symbolisée sur la figure 4 par la loi de Moore) est plus lente que la diminution des coûts de séquençage. En conséquence, la quantité de données nouvellement créées demande de plus en plus de temps pour être traitée, malgré l'augmentation de la puissance informatique.

1.2.3 Assemblage de génomes

Les NGS ne fournissent pas la séquence complète d'un génome, mais seulement un très grand nombre de courtes séquences issues de ce même génome. Pour un génome de t paires de bases, on obtient un grand nombre de séquences pour un total de $x \times t$ paires de bases. Ce x est appelé la *couverture* et est généralement de l'ordre de quelques dizaines. Un séquençage à 10x contient donc, en moyenne, 10 fois chaque paire de bases du génome. Les séquences obtenues se chevauchent partiellement. On essaie alors de reconstruire le puzzle à partir de ces chevauchements, pour obtenir le génome initial. Cette reconstruction est nommée assemblage.

L'assemblage consiste ainsi à attacher ensemble les séquences d'ADN provenant d'un même génome. Le but est de repérer les régions des séquences qui se chevauchent puis de coller ensemble de telles séquences sur la région chevauchante. La séquence obtenue lors de la fusion, appelée *contig*, est plus grande que les séquences de base. Plusieurs contigs non chevauchants peuvent être liés ensemble dans un "*scaffold*". De proche en proche, on essaie de reconstruire le génome, ou le chromosome séquencé, au complet.

On peut voir ce problème comme prendre plusieurs copies d'un même livre et découper toutes les pages en petits morceaux de phrases. Il s'agit alors de reconstruire le livre juste en regardant les petits morceaux obtenus. D'un point de vue informatique, cette tâche est ardue. Il y a probablement des phrases du livre initial qui apparaissent plusieurs fois dans l'ouvrage et il est donc difficile de remettre tous les morceaux dans le bon ordre. De plus, les séquenceurs ne sont pas fiables à 100% et produisent quelques erreurs de séquençage (insertions, délétions ou substitutions), de l'ordre de 0,1% à 1%. De nombreux outils et méthodes d'assemblage ont été développés, tels celera [28], Newbler [29], SOAPdenovo [30], velvet [31], etc.

Plus un génome est composé d'un grand nombre de paires de bases et plus des régions du génome sont répétées, plus la complexité de l'assemblage est importante. Par exemple, un génome bactérien est assez petit (quelques millions de paires de bases) et contient généralement peu de zones répétées. L'assemblage de ce type de génome est assez aisé. Par contre, les génomes eucaryotes, tel celui de l'humain, sont beaucoup plus grands (plusieurs milliards de paires de bases) et contiennent de larges portions répétées. Dès lors, l'assemblage est beaucoup plus difficile.

La plupart des assembleurs nécessitent une grande quantité de mémoire pour fonctionner. Par exemple, SOAPdenovo a besoin de 5 Go pour de petits génomes, mais monte à 150 Go pour un génome humain. Un assembleur récent, Minia [32], permet toutefois d'assembler un génome humain en utilisant moins de 6 Go de mémoire.

1.3 DE NOMBREUSES ESPÈCES NON CULTIVABLES

La génomique dispose de méthodes et d'outils bien établis. La quantité de données séquencées pousse ces outils et méthodes à sans cesse évoluer. Malgré tout, la génomique rencontre des problèmes. Par exemple, certains organismes ont un génome immense. Le cas extrême est une amibe, *Polychaos dubium,* et ses 675 milliards de paires de bases, soit près de 200 fois le génome humain. De tels génomes sont très difficiles à assembler.

Pour séquencer un microorganisme, il est essentiel de le cultiver en laboratoire, ne serait-ce que pour l'isoler et ainsi ne séquencer que lui. Or, beaucoup de microorganismes ne survivent pas en milieu contrôlé : il est donc impossible d'isoler et de séquencer ces espèces, et la génomique n'est alors pas capable d'étudier en profondeur ces organismes. On sait en effet depuis longtemps que de nombreux organismes ne sont pas cultivables en laboratoire. Par exemple, on estime que seul 0,001% à 0,1% des bactéries présentes dans l'eau de mer peuvent effectivement vivre en milieu contrôlé [5]. Plusieurs raisons existent pour expliquer ces difficultés. Une raison simple est l'incapacité à reproduire fidèlement le milieu naturel de ces organismes : un élément clé à leur survie peut manquer. Une autre raison vient du fait que des microorganismes vivent en relation symbiotique ou parasitique avec d'autre organismes [33]. Cultiver ces espèces consisterait alors à cultiver un écosystème de microorganismes.

1.3.1 De la génomique à la métagénomique

Si on ne peut pas cultiver un grand nombre de microorganismes, il est tout de même possible de les étudier, directement dans leur milieu. Il est alors impossible d'isoler une seule espèce des autres et la génomique classique ne peut être utilisée. Comme lors du passage de la génétique à la génomique, on fait face ici à une différence d'échelle. La génétique étudie les gènes d'un organisme, la génomique, le génome d'un organisme, ici on travaille sur un ensemble de génomes inconnus : on parle alors de métagénomique. Historiquement, les première études métagénomiques remontent à 1985 et 1986 [34, 35] mais le terme "métagénomique" fut utilisé pour la première fois en 1998, par Handelsman *et al.* [36].

La métagénomique vise à étudier le contenu génétique d'un échantillon provenant d'un environnement naturel. Par exemple, on prélève un litre d'eau de mer, on récupère l'ensemble de l'ADN présent dans cette eau et on séquence cet ADN. Il n'est ainsi pas nécessaire de pouvoir cultiver les espèces. Par contre, il n'est pas possible de savoir directement quelles espèces ont ainsi été séquencées. La métagénomique pose ainsi de nouvelles problématiques :

A. en terme biologique tout d'abord : combien d'espèces y-a-t'il dans cet échantillon ? Quelles sont les relations entre ces espèces ? Quelles protéines sont sécrétées dans le milieu ? Qui sécrète quoi ? Toutes ces questions trouvent, à différents degrés, leurs réponses, par différentes approches introduites dans la sous-section suivante ;

B. en terme informatique ensuite : comment traiter ces données ? La masse de données peut être de l'ordre de plusieurs dizaines à centaines de gigaoctets pour un seul échantillon. Le projet Tara oceans, introduit sous-section 2.2.4, génère plus de 2000 échantillons, pour un volume de données estimé à plus d'un pétaoctet. Traiter une telle quantité de données demande des algorithmes et des structures de données adaptés.

La métagénomique peut utiliser quelques outils de génomique, mais la plupart des questions soulevées par cette discipline récente nécessitent de réinventer les outils de génomiques, adaptés à ces nouveaux problèmes.

1.3.2 Différentes familles de métagénomique

Comme expliqué précédemment, la métagénomique a différentes facettes et plusieurs problématiques se posent. Cette discipline, bien que récente, propose différentes approches. Dans le cadre de la présente thèse, nous proposons une classification des études métagénomiques en quatre familles distinctes : ciblée, quantitative, fonctionnelle et comparative.

Les analyses ciblées se caractérisent par la recherche de certains organismes ou certains gènes spécifiques présents dans la communauté microbienne analysée. Le but est généralement d'étudier la composition d'un milieu mais aussi de comprendre les liens de parentés existants entre plusieurs milieux. La métagénomique ciblée cherche à répondre à la question : "Qui est présent dans cet échantillon ?" et est introduite plus en détail sous-section 2.1.1.

La métagénomique quantitative cherche à identifier le plus possible d'espèces présentes dans un milieu, puis à estimer la proportion de ces espèces. Ce type d'études peut servir, par exemple, à trouver des liens entre la diversité et la quantité de différents pathogènes chez un individu et une maladie. La métagénomique quantitative cherche à répondre à la question : "Combien sont-ils dans cet échantillon et en quelle proportion ?" et est présentée sous-section 2.1.2.

Les études de métagénomique fonctionnelle se concentrent sur des fonctions d'intérêt exprimées dans un milieu, soit pour comparer différents milieux, soit pour rechercher de nouvelles molécules. Ces études peuvent conduire à la découverte de nouveaux gènes de résistance aux antibiotiques, de nouvelles protéines pharmaceutiques, etc. La métagénomique fonctionnelle cherche à répondre à la question : "Quelles fonctions se déroulent dans cet échantillon ?" et est analysée sous-section 2.1.3.

Finalement, la métagénomique comparative a pour but de comparer différents métagénomes. Cette comparaison peut être effectuée grâce aux autres familles de métagénomique, ou directement sur la base des séquences contenues dans les métagénomes. Ces études mettent à jour, par exemple, des relations entre les conditions physico-chimiques et la diversité microbienne d'un milieu. D'un point de vue médical, des différences entre un individu sain et un individu malade peuvent ainsi être trouvées. La métagénomique comparative cherche à répondre à la question : "Quels sont les différences et les points communs entre ces échantillons ?" et est détaillée sous-section 2.1.4.

Les frontières entre les quatre famille de métagénomique introduites ici sont parfois floues. Des problématiques complexes nécessitent de passer par certaines phases plus quantitatives ou fonctionnelles, avant, par exemple, de mener des analyses comparatives. Mais dans tous les cas, l'objet d'étude est le même : un milieu riche et complexe, composé d'une grande quantité de microorganismes majoritairement inconnus.

1.3.3 Assemblage de métagénomes

Pouvoir assembler un métagénome permettrait d'obtenir tous les génomes présents dans un échantillon. Tous les assembleurs couramment utilisés en génomique sont dédiés à l'assemblage d'un unique génome, ayant une bonne couverture relativement uniforme. Or, en métagénomique, la couverture des différentes espèces est très inégale, et de nombreux organismes ont une faible couverture. Les logiciels d'assemblage ne fonctionnent donc pas correctement sur les données métagénomiques. De plus, des génomes d'espèces proches partagent une partie de leur ADN. Dès lors, les logiciels peuvent fusionner deux séquences provenant d'espèces différentes et créent alors des séquences chimériques [37].

Comme expliqué précédemment, lors de l'assemblage d'un génome classique, les parties répétées de l'ADN posent problème. Or, en métagénomique, plusieurs individus de la même espèces sont présents dans l'échantillon. Dès lors, des répétitions peuvent être de la taille

de ces génomes. Avec des métagénomes très simples, contenant quelques espèces seulement et bien couvertes, les logiciels d'assemblage classiques donnent de bons résultats. Sur des métagénomes beaucoup plus complexes, plusieurs études utilisent ces logiciels d'assemblage, non pas pour reconstituer des génomes complets, mais pour créer des contigs, plus grands que les séquences initiales [38]. Avec ces contigs, on obtient généralement des informations plus précises sur les fonctions ou les gènes présents dans l'échantillon qu'avec les courtes séquences initiales.

Dans ce sens, le logiciel Metavelvet [39] a été construit, sur la base de l'assembleur velvet, afin de maximiser la taille des contigs générés. Les résultats de cet outil montrent que les contigs qu'il génère sont de meilleure qualité et plus longs que ceux produits par les assembleurs classiques. Ainsi, les analyses à partir de ces contigs sont plus précises et le contenu taxonomique des métagénomes est mieux décrit.

L'assemblage de métagénomes n'en est qu'à ses débuts. Dans un récente étude de mai 2013 [40], Albertsen *et al.* ont montré qu'il est possible d'assembler correctement plusieurs génomes présents dans un métagénome. Pour cela, ils ont échantillonné de deux manières différentes un même milieu. Ils ont alors assemblé chacun des deux échantillons pour obtenir des contigs. En alignant l'ensemble des séquences d'un métagénome sur ses contigs, ils ont estimé l'abondance de chacun des contigs. Cette abondance permet de regrouper les contigs avec une couverture comparable dans chacun des deux métagénomes. Par exemple, un ensemble E de contigs est faiblement couvert dans le métagénome A mais fortement couvert dans le métagénome B. Ces contigs sont regroupés ensemble car provenant probablement d'une même espèces, ou tout du moins d'un petit nombre d'espèces. La dernière étape consiste à relancer un assemblage mais en utilisant seulement les contigs d'un groupe. En utilisant cette méthode, Albertsen *et al.* ont pu assembler 12 génomes provenant d'un même milieu.

L'assemblage de métagénomes est un domaine prometteur. Mais, pour le moment, les résultats ne sont pas suffisants pour se baser uniquement sur cette technique.

1.4 CONCLUSION

Comme lors du passage de la génétique à la génomique, la métagénomique consiste en un changement d'échelle. L'objet d'étude n'est plus le génome d'un organisme mais de nombreux génomes provenant d'un même milieu. Ce changement d'échelle s'accompagne d'une modification des problématiques. La plupart des outils génomiques ne sont pas appropriés pour répondre aux nouvelles questions posées par la métagénomique. De plus, les données générées lors d'études métagénomiques atteignent des volumes rarement égalés en biologie. Ainsi, la métagénomique ne peut pas être simplement considérée comme une extension de la génomique et introduit un paradigme différent dans la manière de concevoir les outils bio-informatiques [1]. Des nouvelles structures de données et de nouveaux algorithmes doivent être développés afin d'assurer le passage à cette nouvelle échelle. C'est donc tout un pan de la bio-informatique qui doit être construit ou reconstruit.

Le problème traité durant cette thèse s'inscrit dans la métagénomique comparative. La question biologique est : comment peut-on comparer plusieurs métagénomes en utilisant seulement les séquences qu'ils contiennent, donc sans prendre en compte les connaissances déjà existantes ?

Avant d'essayer d'apporter une réponse informatique à cette problématique, il est nécessaire d'étudier les différentes approches métagénomiques déjà existantes. La spécificité de ces approches fait qu'elles ne sont pas à même de résoudre notre question de manière satisfaisante.

Les méthodes informatiques capables d'apporter une réponse sont généralement trop génériques pour être efficaces ou trop spécifiques pour être utilisées à bon escient. De plus, les

structures de données classiquement utilisées en bio-informatique ne sont pas appropriées pour notre question, compte tenu du volume de données générées par la métagénomique.

L'objectif de ce travail a été de répondre à cette problématique de comparaisons métagénomique. La solution élaborée apporte une réponse satisfaisante, tant du point de vue informatique que biologique.

Les résultats principaux de cette thèse sont :

A. une réflexion sur le paysage métagénomique actuel, menant à une classification en quatre famille ;

B. une nouvelle structure de données très efficace en temps et en mémoire, capable de stocker la présence ou l'absence d'une séquence d'ADN dans un très grand jeu de données ;

C. un nouvel algorithme, nommé COMPAREADS, qui calcule une mesure de similarité entre deux métagénomes basée sur le nombre de séquences similaires partagées par ces deux métagénomes.

Organisation du manuscrit

Le chapitre 2 consiste en un état des connaissances, portant à la fois sur la métagénomique et les différentes techniques existantes, mais aussi sur les structures de données couramment utilisées en bio-informatique. Ce chapitre met en évidence le manque d'outils pour comparer des métagénomes sans recours à des connaissances annexes et l'incapacité des structures de données classiques à répondre à notre problématique.

Le chapitre 3 expose la méthodologie mise au point durant ce doctorat pour comparer deux métagénomes et calculer une mesure de similarité entre eux. Cette similarité est basée sur le nombre de séquences similaires que les deux jeux de données partagent. Une implémentation de cette méthode est proposée, nécessitant une nouvelle structure de données. Cette structure a été créée pour répondre aux besoins spécifiques de la comparaison de grandes quantités de séquences d'ADN.

Dans le chapitre 4, on évalue les performances, en terme de temps, de mémoire et d'exactitude des résultats, de notre nouvelle structure de données face aux structures existantes. On compare ensuite les performances de notre algorithme à différents outils de métagénomique comparative et de génomique. Plusieurs résultats biologiques sur des projets réels de métagénomique sont alors présentés.

Enfin, le chapitre 5 synthétise les contributions présentées dans ce manuscrit, puis introduit plusieurs améliorations et pistes de recherche.

2

ÉTAT DE L'ART

Ce chapitre constitue une introduction aux questions et méthodes apportées par les différentes familles de métagénomique. Après avoir défini de manière générale ce que l'on entend par "métagénomique", nous proposerons quatre familles de métagénomique. Les outils bio-informatiques qui peuvent y être associés sont décrits et analysés. Plusieurs outils qui proposent des approches statistiques à des questions métagénomiques existent. Ils ne sont pas présentés dans cet état de l'art, nous nous concentrons ici sur des approches que l'on peut qualifier d"'orientées séquences". Avec ces approches orientées séquences, il est possible d'obtenir un ensemble de séquences répondant à une question donnée : ces séquences peuvent alors être utilisées par d'autre logiciels.

Suite à cet état des connaissances, certains projets de métagénomique de grande envergure sont présentés. La plupart de ces projets sont aujourd'hui terminés et les conclusions de ces études sont analysées.

En lien avec les problématiques méthodologiques, ce chapitre passe aussi en revue un certain nombre de structures de données couramment employées en bio-informatique pour le traitement de grandes masses de données. Les avantages et inconvénients de ces différentes structures sont étudiés. Finalement, la conclusion nous amène à appréhender le point de départ de ces travaux de thèse qui sont à la rencontre de la métagénomique comparative et des structures de données optimisées : la création d'une nouvelle méthode permettant de comparer efficacement d'énormes jeux de données métagénomiques.

2.1 DIFFÉRENTES FAMILLES MÉTAGÉNOMIQUES

La métagénomique est une discipline récente étudiant le contenu génétique d'un échantillon issu d'un environnement naturel (sol, mer, etc). Les premières études utilisant du matériel génétique directement extrait d'un milieu naturel [34, 35] ont été conduites entre 1985 et 1986. Elles s'intéressaient à l'ARN ribosomique (ARNr). L'étude de Pace *et al.* [41], en 1991, a montré, entre autres, la large distribution dans les océans de bactéries non-caractérisées à l'époque. Le terme "métagénomique" a été introduit plus tard, en 1998, par Handelsman *et al.* [36]. Le protocole utilisé lors de ces premières expériences ressemblait au protocole classiquement utilisé. De nos jours on peut résumer ainsi ce protocole :

A. récupération du matériel biologique directement dans l'environnement (air, sol, eau, etc) ;
B. filtrage pour ne garder que les cellules intéressantes pour l'étude ;
C. lyse (désintégration de la membrane cellulaire) pour récupérer l'ADN de ces cellules. Aucune mise en culture des organismes n'est effectuée. La récupération du matériel génétique est directe ;
D. purification de l'ADN afin de retirer toutes les autres molécules biologiques ;
E. multiplication des séquences d'ADN. À cette étape, il est possible de sélectionner le type d'ADN à étudier mais aussi de s'intéresser à certains gènes seulement (voir Séquençage dirigé par la fonction) ;

f. séquençage de l'ADN grâce à un séquenceur de nouvelle génération (aussi appelé NGS pour *Next-Generation Sequencing*) ;

g. analyse des séquences obtenues.

Il y a deux manières d'obtenir des séquences métagénomiques [42]. La première manière, nommée *séquençage dirigé par la fonction*, consiste à se baser sur **les fonctions exprimées** par un métagénome [43]. La seconde manière, plus récente, est le *séquençage complet*. On cherche ici à obtenir **le plus possible de séquences d'**ADN contenues dans un métagénome.

SÉQUENÇAGE DIRIGÉ PAR LA FONCTION Le séquençage dirigé par la fonction, utilisé dès les débuts de la métagénomique, permet d'obtenir des gènes répondant à une fonction précise. L'ensemble de l'ADN présent dans un échantillon est extrait. L'ADN ainsi récupéré est alors découpé et chaque partie est insérée dans une autre molécule d'ADN capable de se répliquer. On nomme cette seconde molécule un **vecteur**. En se répliquant, le vecteur va aussi répliquer l'ADN étranger qui y est inséré. Les vecteurs sont insérés dans un hôte sain, généralement *Escherichia coli*. Les fragments insérés dans les vecteurs sont alors exprimés par l'hôte. Il devient possible de rechercher la présence d'une activité biologique spécifique, comme la création d'une enzyme donnée ou la résistance à un antibiotique, qui serait encodée dans l'ADN du métagénome. Une stratégie simple pour, par exemple, détecter une résistance à un antibiotique, consiste à cultiver l'hôte en présence de l'antibiotique. Seuls les hôtes contenant une molécule d'ADN exprimant la résistance à l'antibiotique vont pouvoir se développer.

Cette méthode permet d'isoler de nouvelles protéines totalement inconnues à partir de microorganismes actuellement impossibles à cultiver en laboratoire [42]. Mais cette méthode peut engendrer des biais. Un métagénome est un mélange complexe d'un grand nombre de génomes provenant de microorganismes potentiellement très différents. L'hôte utilisé pour exprimer les fragments d'ADN, généralement *E. coli*, peut ne pas avoir la machinerie cellulaire nécessaire pour reconnaître les signaux de transcription ou traduction de certains microorganismes. L'hôte peut donc ne pas arriver à décoder certaines informations génétiques contenues dans le métagénome [44].

SÉQUENÇAGE COMPLET Les récentes avancées des NGS rendent désormais possible le séquençage aléatoire d'une partie de l'ADN présent dans un échantillon métagénomique [42]. Il est important de noter qu'une analyse métagénomique d'un milieu complexe ne représente souvent qu'une fraction de la communauté microbienne présente à un moment donné. Pour être exhaustif, produire un jeu de données représentatif de la communauté microbienne d'un unique gramme de sol terrestre nécessiterait plus de 6000 cycles de fonctionnement (*run*) d'un séquenceur actuel, pour un coût total de plusieurs centaines de millions de dollars [1, 45, 46]. Ainsi, le séquençage complet tel qu'utilisé actuellement ne contient pas l'ensemble de l'ADN d'un milieu, mais les portions d'ADN séquencées ne sont pas choisies, la sélection est aléatoire.

Un tel séquençage permet d'obtenir facilement un très grand nombre de séquences. Pour ce faire, l'ADN extrait et purifié est soumis à une amplification. Cette amplification permet de récupérer suffisamment de matériel génétique pour procéder au séquençage. Une méthode couramment utilisée en biologie moléculaire est la PCR (*Polymerase Chain Reaction*, réaction en chaîne par polymérase). Durant la PCR, l'ADN à amplifier est chauffé pour séparer les deux brins. Des amorces (courtes séquences d'ADN) peuvent alors s'hybrider sur l'ADN à répliquer. Des polymérases interviennent alors. Une polymérase est une enzyme présente dans tout organisme vivant et dont le rôle est de recopier une molécule d'ADN en une autre, normalement identique. Dans la PCR, ces polymérases se fixent sur les amorces et recopient la molécule d'ADN en une nouvelle.

En métagénomique, les PCR sont très utiles pour obtenir très rapidement une grande quantité d'ADN. L'utilisation de PCR permet de récupérer les séquences du métagénome ayant une

homologie forte aux amorces utilisées. Ceci peut permettre de cibler quelles séquences amplifier préférentiellement parmi tout l'ADN disponible. Mais l'amplification par PCR peut empêcher de trouver des séquences totalement nouvelles [44]. Pour remédier à cela, on peut utiliser une MDA (*Multiple Displacement Amplification*, amplification à déplacement multiple). Dans le détail, cette technique est différente de la PCR en plusieurs points. Le plus important est la taille des amorces utilisées. En PCR, les amorces font une vingtaine de nucléotides tandis qu'en MDA on utilise des amorces de six nucléotides. On peut alors cibler pratiquement n'importe quelle séquence. Mais, cette technique engendre aussi différents biais [47, 48, 49].

Ces deux processus ont chacun leurs avantages et inconvénients et doivent être adaptés à l'étude métagénomique menée. Dans les deux cas, à la fin du processus, on récupère de nombreuses portions d'ADN inconnu appartenant à des espèces non cultivables. Dès lors, plusieurs questions peuvent se poser et différents types d'approches métagénomiques existent. Les questions auxquelles on cherche à répondre sont généralement : "Qui est présent dans cet échantillon ?", "Quelles réactions se déroulent dans cet échantillon ?" ou encore "quel organisme est responsable de cette réaction ?" [42].

Nous proposons de regrouper les différentes approches métagénomiques en quatre familles. Plus précisément, on peut rechercher si des organismes connus sont présents dans un métagénome donné ou si des fonctions connues s'y déroulent en faisant de la Métagénomique ciblée. La Métagénomique quantitative cherche à faire un catalogue le plus exhaustif possible des gènes connus présents dans un métagénome. Une fois ce catalogue obtenu, on cherche à estimer l'abondance de chacun des gènes ou des génomes présents dans les échantillons. La Métagénomique fonctionnelle recherche de nouvelles fonctions ou protéines apparaissant dans un métagénome. Enfin, la Métagénomique comparative consiste à comparer plusieurs métagénomes afin, par exemple, de trouver des espèces ou des gènes partagés par ces métagénomes, d'établir une distance entre eux, etc. Cette thèse s'inscrit dans la **métagénomique comparative**.

2.1.1 Métagénomique ciblée

OBJECTIFS La métagénomique ciblée cherche à identifier les membres d'un métagénome ou à tester la présence/absence d'un organisme ou d'un gène dans un métagénome. Le but peut être d'étudier la composition d'une communauté microbienne mais aussi de comprendre les liens de parentés entre plusieurs communautés, d'étudier leurs évolutions et réponses à des stress environnementaux, etc. La métagénomique ciblée permet de s'intéresser à des organismes ou à des gènes présents dans un métagénome même en très faible quantité. Ce qui pourrait être considéré comme un bruit de fond en métagénomique fonctionnelle ou quantitative ne le sera pas si il est spécifiquement ciblé [50].

MÉTHODES Il existe deux méthodes pour réaliser de la métagénomique ciblée. La première consiste à créer des *sondes*, des courtes portions d'ADN ne pouvant s'hybrider qu'avec certaines séquences. Typiquement, si on veut savoir si telle bactérie est présente dans le milieu, on utilise une sonde spécifique capable de ne se fixer qu'à une portion représentative de son génome. Une fois les sondes créées, on les utilise comme amorces dans une PCR. À la fin de la PCR, seules les parties s'hybridant avec les sondes seront récupérées : si la bactérie est présente, on aura de nombreuses copies de ces fractions d'ADN. La seconde technique consiste à tout séquencer puis à rechercher les séquences d'intérêt *in silico*.

Dans, les deux cas, après séquençage, les séquences obtenues sont alignées sur des séquences de références connues. Les séquences recherchées peuvent être des marqueurs, tels les ARN ribosomiques (ARNr) 16s ou 18s. Ces ARN ont la particularité d'exister chez presque

tous les organismes vivants et de tous avoir une partie identique. Les différences sur le reste de la séquence témoignent alors de l'éloignement des espèces : plus deux espèces ont un ARN 16S similaire, plus elles sont proches d'un point de vue évolutif. En ciblant ces ARN, on peut identifier des espèces données présentes dans l'échantillon [51, 52]. Il est aussi possible de rechercher des gènes d'intérêt codant pour une protéine donnée [53]. De même, comme en métagénomique fonctionnelle (voir sous-section 2.1.3), il est possible de rechercher des gènes de résistance aux antibiotiques [54, 55]. À la différence de la métagénomique fonctionnelle, il s'agit ici de savoir si l'échantillon contient des mutations ou des gènes qui sont déjà connus comme étant résistants [56].

La métagénomique ciblée est donc dépendante des connaissances actuelles et une grande partie des microorganismes est encore inconnue. La précision de l'analyse repose en partie sur la qualité et l'exhaustivité des bases de données : les résultats ne sont valables que si la séquence de référence est valide [50]. Or, un certain nombre de génomes dans les bases de données actuelles contiennent des erreurs d'annotation [57]. Un autre biais dû à l'ADN peut aussi se présenter dans des analyses visant à détecter un agent pathogène dans un métagénome. L'ADN retrouvé dans le métagénome peut correspondre à un individu mort : on conclura, à tord, à la présence du pathogène au moment du séquençage.

MOYENS BIO-INFORMATIQUES Plusieurs outils bio-informatiques existent pour l'étude de données de métagénomique ciblée. Dans ce qui suit, un certain nombre de ces outils sont présentés et discutés. Les trois premiers outils, ARB [58], MUSCLE [59] et RDP [60], sont utilisés pour trouver des séquences d'ADN caractéristiques d'un gène, d'une espèce ou d'un ensemble d'espèces proches d'un point de vue évolutif. Ces séquences sont utilisées comme sondes dans une PCR avant séquençage, ou comme séquence à rechercher *in silico* après séquençage. Un autre logiciel, TriageTools [61], teste la présence de séquences cibles dans un ensemble de séquences métagénomiques.

L'outil ARB [58] combine plusieurs logiciels interagissant les uns avec les autres. Au cœur de l'outil, une base de données regroupe des organismes, des gènes et des produits de gènes (ARN ou protéines). Ces données proviennent de différentes sources publiques comme l'EBI (*European Bioinformatics Institute*) ou GenBank [62], une des plus grosses banques de séquences d'ADN. Plusieurs outils sont disponibles pour interroger cette base sous différents angles. La fonction d'ARB la plus intéressante en métagénomique ciblée est la conception de sondes. Une sonde recrutant plusieurs organismes doit être suffisamment spécifique pour ne s'hybrider qu'aux organismes ciblés mais suffisamment générique pour tous les recruter. Les programmes "Probe Design" et "Probe Match", contenus dans ARB, identifient de courtes séquences spécifiques à un unique organisme. Dans le cas d'une sonde pouvant s'hybrider à plusieurs organismes proches, Probe Design récupère les sondes spécifiques à chacun des organismes et procède à des alignements multiples pour trouver les régions les plus proches entre ces sondes. Des séquences consensus sont dérivées de ces alignements puis évaluées par Probe Match. Pour cela, Probe Match effectue des alignements locaux entre ces sondes potentielles et les séquences les plus similaires trouvées dans l'ensemble de la base de donnée d'ARB. Les meilleurs résultats sont retournés et permettent de confirmer la spécificité des sondes : si une sonde recrute seulement les organismes choisis et les recrute tous, elle est sélectionnée. ARB est un outil performant de conception des sondes pour recruter des séquences codantes ou des organismes connus et référencés.

Un autre logiciel, MUSCLE [59], est parfois utilisé pour aider à créer des sondes [50]. Ce programme est conçu pour réaliser des alignements multiples de séquences protéiques ou nucléiques. Il n'est pas spécialement dédié à la métagénomique, mais il peut être utilisé pour la conception de sondes afin de cibler, par exemple, plusieurs espèces d'un même genre par-

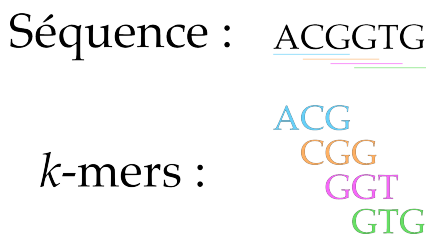


FIGURE 5: Représentation d'une séquence d'ADN et de ses k-mers correspondant pour $k = 3$. Une séquence de longueur l est composée de $l - k - 1$ k-mers.

FIGURE 6: Comptage des k-mers partagés entre des séquences proches ou éloignées pour $k = 3$. Les deux premières séquences sont proches et partagent trois k-mers (en bleu). La dernière séquence est éloignée des deux premières. Elle ne partage qu'un seul k-mer avec la première séquence (en orange).

tageant une région conservée [63]. MUSCLE commence par calculer une distance entre toutes les séquences prises deux à deux en utilisant des *k-mers*. Un k-mer est un mot de taille k. On peut découper une séquence s, composées de n nucléotides, en $n - (k - 1)$ k-mers consécutifs (voir figure 5. De plus, une définition formelle de *k-mer* figure sous-section 3.1.2-B). Partant du principe que deux séquences proches partagent plus de k-mers qu'attendu par hasard (voir figure 6), on peut rapidement attribuer un score de similarité entre chaque paire de séquences sans avoir à procéder à un alignement multiple. Ce score de similarité entre séquences permet de créer une matrice de distances entre chaque paire de séquences. Un clustering par UPGMA (*Unweighted Pair Group Method with Arithmetic mean*) est appliqué sur cette matrice de distances. Ce type de clustering permet de créer un arbre phylogénétique regroupant les séquences les plus proches sur une même branche.

Un premier alignement multiple est alors progressivement construit à partir de cet arbre. Un alignement multiple est réalisé entre les séquences portées par une même branche. Une séquence consensus est produite pour cet alignement multiple. Puis, en remontant dans l'arbre, les séquences consensus sont alignées entre elles à travers un nouvel alignement multiple. La séquence consensus alors générée est à son tour alignée avec les autres séquences consensus du même niveau, et ce jusqu'à atteindre la racine de l'arbre. Ce processus à l'avantage d'être beaucoup plus rapide qu'un alignement multiple direct entre toutes les séquences : chaque alignement à réaliser ici se passe avec une petite quantité de séquences.

La distance entre les séquences, obtenue à partir des k-mers communs, est approximative et l'alignement multiple peut être sous-optimal. Pour améliorer l'alignement, MUSCLE utilise la distance de KIMURA [64]. Cette distance est plus précise mais nécessite d'avoir des séquences alignées. La distance de KIMURA consiste à quantifier la distance évolutive entre deux séquences en pénalisant les mésappariements (ou *mismatch*). Une nouvelle matrice de distances entre toutes les séquences est ainsi obtenue. Cette nouvelle matrice de distances est de nouveau convertie en arbre après clustering par UPGMA. Comme pour le premier arbre, un alignement multiple est calculé. Cet alignement est le résultat de MUSCLE.

Dans le cadre de la métagénomique ciblée, il est possible d'aligner ainsi plusieurs séquences venant d'organismes différents. La séquence consensus finale sert alors de sonde. MUSCLE à l'avantage d'être un des logiciels d'alignement multiple les plus rapides. Mais, bien que MUSCLE puisse travailler sur n'importe quelle séquence, la construction de sondes repose uniquement sur les connaissances actuelles et empêche de travailler avec des séquences totalement nouvelles.

RDP [60] est une base de données de séquences d'ARNr d'archées et de bactéries. Une des particularités de RDP est que les séquences qu'elle contient sont déjà alignées. Les alignements réalisés sont fait en utilisant *Infernal Aligner* [65]. Cet aligneur ne fait pas un alignement de séquences classique : il se base sur la structure secondaire des ARN. La structure secondaire représente partiellement la structures tridimensionnelle de certaines parties d'une bio-molécule. Cette structure est plus conservée que la séquence : en effet, la structure d'une molécule est intimement liée à sa fonction. Si deux séquences sont différentes mais qu'elles ont la même structure, leur fonction est probablement proche. Ainsi, il est possible de créer des sondes répondant à une fonction particulière plutôt que correspondant seulement à quelques séquences données.

Comme pour les deux logiciels précédents, la création de sondes se base sur les connaissances actuelles. Toutefois, utiliser des informations tridimensionnelles peut être plus pertinent pour cibler une fonction précise.

Un outil très récent, TriageTools [61], recherche une liste de séquences cibles connues parmi un ensemble de séquences métagénomiques. L'idée est de comparer deux séquences en comptant le nombre de k-mers (voir figures 5 et 6), qu'elles partagent. Lors de la première étape, tous les k-mers des séquences métagénomiques sont récupérés. Il y a au plus 4^k k-mers différents. Chaque k-mer est alors transformé est une séquence binaire. Pour cela, chaque nucléotide est codé sur deux bits : par exemple, A sera codé par 00, C par 01, G par 10 et T par 11. À chaque k-mer correspond donc une séquence unique de $2k$ bits. La seconde étape consiste à créer un tableau associatif de 4^k bits dont tous les bits sont à 0. Chaque séquence binaire de k-mer pointe donc vers une unique case de ce tableau, et le bit à cette case est mis à 1. Dans la troisième étape, toutes les séquences cibles sont traitées une par une. La séquence est découpée en k-mers qui sont convertis en séquences binaires. Pour chacune de ces séquences binaires, on regarde alors dans le tableau si elle pointe vers une case à 0 ou à 1. Si la case est à 1, c'est que le k-mer correspondant est présent dans le métagénome. Ainsi, un score est attribué à chaque séquence cibles sur la base du nombre de k-mers partagés avec les séquences métagénomiques. Si ce nombre de k-mers dépasse un seuil H, la séquence cible est dite présente parmi les séquences métagénomiques.

Cet outil à l'avantage d'être rapide, sa complexité en temps est en $O(NL)$, où N est le nombre total de séquences à analyser et L est la taille de ces séquences. Dans cette méthode, la taille occupée en mémoire n'est pas dépendante du nombre de séquences à traiter, mais de la taille des k-mers. Un désavantage de cette méthode est qu'elle peut identifier à tord une séquence comme similaire à une autre : on nomme ce phénomène faux positif. En effet, une séquence cible peut avoir tous ses k-mers présents dans l'index, mais sur des séquences métagénomiques différentes. La probabilité p d'obtenir un faux positif est estimé approximativement à partir de l'équation 1, où ρ est le densité de remplissage du tableau associatif, R est la taille de la séquence cible, k est la taille des k-mers, H est le seuil de détection et T est le

nombre de nucléotides total des séquences métagénomiques.

$$\begin{aligned} p &\propto 2\binom{R/k}{H/k}\rho^{H/k} \\ &= 2\left(\frac{(R/k)!}{(H/k)!((R-H)/k)!}\right)4^{-H}T^{H/k} \end{aligned} \tag{1}$$

L'ensemble des approches décrites ici permettent de travailler après un séquençage complet. Les séquences peuvent donc contenir des organismes inconnus. Si ces logiciels ont tous besoin de séquences cibles connues, il est possible de réutiliser le jeu de données métagénomiques en fonction de l'avancée des connaissances. Contrairement à un séquençage ciblé par PCR, on peut donc rechercher des gènes ou des génomes inconnus lors du séquençage complet.

CONCLUSION La métagénomique ciblée est un très bon moyen d'obtenir des informations sur une communauté microbienne. Un des avantages de cette discipline est qu'il est possible de s'intéresser à des organismes ou à des gènes présents en très faible quantité dans un métagénome [50]. Outre tester la présence d'un organisme dans une communauté, la métagénomique ciblée a permis d'étudier spécifiquement certains pathogènes difficiles à cultiver en laboratoire [66]. De même, cibler particulièrement quelques gènes ou espèces dans un milieu permet de comparer plusieurs conditions ou d'étudier l'évolution de différents milieux à des stress environnementaux [67]. Par contre, la métagénomique ciblée se base uniquement sur les données actuellement présentes dans les bases de données. Outre les erreurs que contiennent ces bases [57], une approche par métagénomique ciblée empêche la découverte de protéines ou génomes totalement nouveaux. De plus, en ciblant un gène particulier, on ne peut savoir exactement quelles espèces présentes dans le métagénome expriment ce gène. Finalement, en concentrant l'expérience sur quelques cibles seulement, il est parfois difficile de connaitre la quantité du métagénome impliquée dans ces cibles. Pour cette dernière problématique, la métagénomique quantitative apporte des réponses.

2.1.2 Métagénomique quantitative

OBJECTIFS Le but de la métagénomique quantitative est de faire un catalogue de références des espèces ou des gènes connus qui sont présents dans un ou plusieurs métagénomes, puis de quantifier chacun de ces gènes ou espèces dans chaque échantillon [68]. Dans le cas d'études cliniques, cette quantification permet la recherche d'associations entre des gènes ou des espèces, et des maladies [69]. On peut aussi comparer plusieurs échantillons sur la base des gènes ou des espèces qu'ils partagent. Il est possible, comme en métagénomique ciblée, de s'intéresser à des séquences particulières, tels les ARNr, pour estimer la quantité de génomes contenus dans un métagénome. De même, la métagénomique quantitative peut permettre d'estimer la quantité de microorganismes présents dans un échantillon qui sont capables de réaliser une fonction biologique particulière, par exemple la photosynthèse [70].

MÉTHODES Les séquences obtenues après un séquençage complet sont alignées sur les données présentes en base de données [71]. Les séquences des métagénomes ayant une forte similarité à des séquences référencées sont conservées et héritent des informations de leurs homologues connus (organisme d'appartenance, rôle biologique, etc). Le catalogue de séquences annotées ainsi obtenu sert alors de base de références. Toutes les séquences des métagénomes étudiés sont projetées contre cette base de références, à nouveau par alignement. On peut ainsi avoir une idée de l'abondance de différents gènes, fonctions ou génomes contenus dans

les échantillons. Le résultat, à la sortie de ce processus, peut être une matrice de comptage contenant sur les lignes les échantillons métagénomiques, et sur les colonnes les espèces ou gènes identifiés dans au moins un des métagénomes.

Une autre méthode, utilisée entre autres dans le projet METAHIT [69], consiste à procéder en premier lieu à un assemblage de chaque métagénome à étudier. Les séquences n'ayant pas été assemblées, tous métagénomes confondus, sont regroupées et l'assemblage est relancé sur ce nouvel ensemble. Tous les contigs obtenus à la fin de ces deux assemblages *de novo* sont annotés comme précédemment expliqué.

La métagénomique quantitative repose entièrement sur les données présentes en base de données. Comme pour la métagénomique ciblée, les analyses peuvent donc être biaisées, d'une part car on se limite aux données connues, et d'autre part car un certain nombre de génomes dans les bases de données actuelles contiennent des erreurs d'annotation [57]. Les paramètres des logiciels d'alignement utilisés peuvent aussi être primordiaux : trop laxistes, certaines séquences vont être mal annotées, mais trop restrictifs, des séquences avec quelques mutations seront ignorées.

MOYENS BIO-INFORMATIQUES De nombreux logiciels permettent d'annoter un jeu de séquences métagénomique. Entre autres, IMG/M [72], MG-RAST [73] et MEGAN [74] sont connus pour répondre à cette problématique. Ces logiciels sont aussi largement utilisés en métagénomique fonctionnelle ; ils seront donc analysés dans la sous-section 2.1.3. Dans le cadre de la métagénomique quantitative, on s'intéresse à deux programmes plus spécifiques, smashcommunity [75] et GASIC [68]. Ces deux programmes sont en effet dédiés à la quantification des espèces présentes dans un métagénome puis à l'estimation de leur abondance respective.

Smashcommunity [75] est un logiciel qui créée une représentation d'un métagénome. Cette représentation, nommée **profile phylogénétique quantitatif**, est composée des différentes espèces retrouvées dans ce métagénome et de leur abondance, les espèces étant placées sur un arbre phylogénétique.

La caractérisation taxonomique des espèces est réalisée par deux méthodes différentes. La première utilise BLAST [22] et récupère, pour chaque séquence, l'espèce la plus proche dans les bases de données publiques. La classification taxonomique de l'espèce est aussi récupérée dans ces bases. Ces informations taxonomiques servent à construire la branche de l'arbre phylogénétique correspondant à la séquence. La seconde manière de caractériser les espèces du métagénome vise à identifier les ARN 16S présents dans l'échantillon puis à les classifier en utilisant la base de données RDP [60] et l'outil de classification qui y est intégré [76].

Le profil phylogénétique quantitatif est alors raffiné en calculant l'abondance relative de chaque espèce dans le métagénome. Pour cela, on compte le nombre de séquences appartenant à une espèce donnée, et ce nombre est pondéré par la taille du génome de l'espèce ; une espèce avec un petit génome mais un grand nombre de séquences dans l'échantillon est donc vue comme très représentée dans le métagénome. De même, pour les ARN 16S, le nombre de copies d'un ARNr donné reflète la représentation de l'espèce correspondante dans le métagénome.

À partir de ces profils, smashcommunity peut effectuer des comparaisons multiples entre plusieurs métagénomes. Une clusterisation hiérarchique des profils est réalisée sur la base des espèces présentes dans l'arbre, mais aussi sur leur abondance relative.

Smashcommunity permet donc d'analyser plusieurs métagénomes, à la fois individuellement et simultanément. Individuellement, il délivre un arbre phylogénétique des espèces présentes dans un échantillon et indique leur abondance. L'analyse comparative des métagénome peut renseigner sur les liens de parenté entre les métagénomes. Un des désavantages principaux de smashcommunity est qu'il ne fonctionne pas avec toutes les technologies de séquençage actuelles : il ne fonctionne qu'avec les séquences produites par sanger ou par ROCHE 454. Or, de nombreuses études sont actuellement menées en utilisant les technologies ILLUMINA

ou ION TORRENT.

Pour quantifier un métagénome, GASIC [68] utilise deux notions d'abondance distinctes. La première, **l'abondance observée**, est utilisée lors de la première phase du logiciel comme mesure préliminaire. GASIC raffine alors cette mesure pour obtenir **l'abondance estimée**, plus précise.

La première phase consiste à aligner toutes les séquences du métagénome sur toutes les espèces connues présentes en base de données. Ensuite, le logiciel compte les séquences qui s'alignent sur les références. À l'inverse de smashcommunity, GASIC ne prend pas en compte seulement l'espèce sur laquelle la séquence s'aligne le mieux : si une séquence s'aligne de manière convenable (suivant les paramètres d'alignement) sur plusieurs espèces, elle sera comptée une fois par espèce. Il n'y a pas de restriction pour obtenir une unique espèce par séquence.

La seconde étape permet de donner une estimation de similarité. Des séquences sont générées à partir des génomes de chaque espèce retenue lors de la première étape en tenant compte des spécificités de la technologie de séquençage utilisée (taux d'erreur, taille des séquences générées, etc). Les séquences obtenues sont donc semblables, en terme d'erreur de séquençage et de longueur, aux séquences du métagénome à étudier. La première étape est appliquée à ces séquences. Si une séquence d'un génome A s'aligne sur un génome B, c'est que ces deux génomes sont assez proches, et donc, certaines séquences du métagénome ont probablement fait cette même erreur. Cette étape permet de construire une matrice de similarité entre les génomes de référence. Grâce à cette matrice, la troisième étape permet de corriger le nombre de séquences alignées par génome. L'abondance observée pour une espèce donnée est un mélange de l'abondance réelle, pondérée par les probabilités de mésalignements estimés issus de la matrice.

Une quatrième étape utilise du bootstrap afin de rendre l'estimation de l'abondance plus stable. La technique du bootstrap consiste à répéter un processus aléatoire plusieurs fois, afin d'obtenir une valeur plus stable. Ici, la création des séquences à partir des génomes de référence, la création de la matrice puis la correction de l'abondance sont ainsi menées plusieurs fois. L'abondance estimée est finalement créée à partir de ce bootstrap et permet d'avoir une mesure stable.

La dernière étape de GASIC consiste en une vérification par l'utilisateur de la qualité du processus. Le logiciel indique le nombre de séquences alignées par génome et la couverture de chaque génome par les séquences du métagénome. En outre, GASIC fournit un histogramme de la couverture de chaque génome. Grâce à ces informations, on peut repérer des schémas significatifs d'un problème. Typiquement : un génome contenant de grandes parties mal couvertes par les séquences du métagénome et ayant une distribution semblable à une loi de POISSON sur les régions fortement couvertes peut indiquer que le génome utilisé n'est pas le bon, mais probablement celui d'une espèce proche.

GASIC est original de par son fonctionnement. L'abondance estimée et corrigée au fur et à mesure du logiciel apporte un bon indice dans le cadre de la métagénomique quantitative. Mais, les multiples générations de séquences pour chaque génome retrouvé dans le métagénome peuvent s'avérer très coûteuses en temps pour des échantillons contenant un très grand nombre d'espèces. Pour un échantillon contenant N génomes, la complexité en temps est en $O(N^2)$. Dans un tel cas de figure, les auteurs recommandent d'estimer la similarité entre échantillons en utilisant d'abord des méthodes beaucoup plus rapides mais moins précises, comme des approches basées sur un comptage de k-mers [77]. Alors, seuls les génomes ayant une similarité de plus de 1% sont traités par GASIC.

CONCLUSION Comme pour la métagénomique ciblée, il est essentiel en métagénomique quantitative d'avoir des espèces de référence bien connues. En effet, l'étape d'alignement

des séquences du métagénome sur les bases de données est déterminante pour la suite de l'étude. Dans un cas idéal, chaque séquence s'aligne sur un seul génome de référence. Mais dans la réalité, la référence est parfois éloignée d'un point de vue évolutif de l'espèce présente dans l'échantillon. Parfois même, aucune espèce proche n'existe, et les séquences ne s'alignent sur aucun génome connu. Par exemple, dans le projet METAHIT [69], seules 31,0 à 48,8% des séquences obtenues, selon les jeux des données, s'alignent sur des génomes bactériens connus constitutifs de la flore intestinale humaine, et 7,6 à 21,2% à d'autres génomes bactériens connus.

Un autre cas problématique en métagénomique quantitative arrive lorsque les espèces constitutives d'un métagénome sont trop proches. Les séquences sont alors alignées sur plusieurs références, et il peut être très difficile d'évaluer correctement l'abondance de chaque espèce. Il est donc nécessaire d'avoir des génomes de références très représentatifs de l'écosystème étudié. Faire de la métagénomique quantitative sur un milieu très peu connu se révèle donc particulièrement ardu.

Une problématique parallèle à la métagénomique quantitative consiste à rechercher les fonctions d'intérêt exprimées dans un métagénome. La sous-section suivante se concentre sur cette branche de la métagénomique.

2.1.3 Métagénomique fonctionnelle

OBJECTIFS La métagénomique fonctionnelle se concentre sur des fonctions d'intérêt exprimées dans un métagénome. Li *et al.* définissent la métagénomique fonctionnelle comme étant "la caractérisation des membres fonctionnels clés d'un microbiome qui influent le plus sur le métabolisme de l'hôte et donc sur sa santé." [78]. Plutôt que de rechercher quelles espèces sont présentes dans l'hôte, cette approche vise à faire un lien entre le métabolisme de l'hôte et les gènes exprimés par le métagénome. Ceci permet une meilleure compréhension des mécanismes inhérents à la bonne santé de l'hôte et de ses réponses à des traitement médicaux.

Cependant, une définition plus large permet d'englober dans la métagénomique fonctionnelle de nombreuses études visant à rechercher des fonctions d'intérêt telles de nouvelles enzymes sécrétées dans l'environnement [44, 79, 80], des gènes de résistance aux antibiotiques [81, 55], de nouvelles cibles pharmaceutiques, etc.

MÉTHODE Comme pour la métagénomique comparative ou ciblée, après séquençage les séquences sont comparées aux données connues et présentes en bases de données. De nombreux logiciels existent afin d'attribuer une fonction à une séquence en utilisant les bases de données génomiques. Un des moyens les plus utilisés consiste à rechercher des similarités entre une séquence inconnue et des séquences déjà annotées en base de données en utilisant BLAST [22]. Si la séquence inconnue ressemble à une séquence bien étudiée d'un point de vue fonctionnel, on peut inférer la fonction de la séquence requête.

Mais, contrairement à la métagénomique ciblée, on s'intéresse moins aux séquences elles-mêmes qu'aux **fonctions** de ces séquences : la métagénomique ciblée cherche à savoir si tels organismes ou séquences connus sont présents dans l'échantillon, la métagénomique fonctionnelle, quand à elle, cherche à inférer les fonctions qui se déroulent dans le milieu prélevé. La différence étant assez ténue, beaucoup d'outils de métagénomique fonctionnelle sont utilisables en métagénomique ciblée ou quantitative.

MOYENS BIO-INFORMATIQUES De nombreuses bases de données contenant des informations fonctionnelles sur des séquences ou des protéines sont disponibles. Parmi les plus connues, on peut citer UNIPROT [82] qui regroupe plusieurs bases de données protéiques et donne des informations sur la séquence, la structure et la fonction de diverses protéines. KEGG

[83] est un ensemble de bases de données regroupant des réseaux métaboliques, des génomes complets, des enzymes, des gènes et des protéines. Kegg permet ainsi une annotation à différents niveaux. Une séquence peut donc être rattachée à la fois à une protéine ou à une fonction donnée, mais aussi placée dans un arbre phylogénétique expliquant les liens de parenté avec d'autres organismes. Les bases de données pfam [84] et tigrfams [85] utilisent le concept de familles protéiques. Une famille protéique contient un ensemble de protéines codées par des gènes issus d'un même ancêtre commun, donc avec des fonctions proches. Ces deux bases de données permettent d'annoter les gènes ou les génomes où sont retrouvées ces protéines. La base de données eggnog [86] sert à obtenir des annotations fonctionnelles sur des groupes de gènes orthologues, des gènes issus d'un même ancêtre commun et ayant divergé dans différentes espèces. Une autre base de données, cog [87], contient des clusters de groupes de protéines. Chaque groupe de protéines est composé de plusieurs protéines orthologues. Chaque cluster est associé à différents niveaux phylogénétiques. Une séquence retrouvée dans un cluster peut ainsi être immédiatement placée sur un arbre phylogénétique.

Toutes ces bases de données sont partiellement redondantes les unes avec les autres, mais aucune ne peut être exhaustive et couvrir totalement l'ensemble des fonctions et processus biologiques. Dès lors, plusieurs algorithmes et logiciels ont été créés afin d'interroger simultanément plusieurs de ces bases de données, puis, dans un second temps, de confronter les différents résultats obtenus. Les deux plus célèbres systèmes [88] sont mg-rast [73] et img/m [72]. Deux autres logiciels méritent d'être cités. Le premier, megan [74], est largement utilisé pour étudier le contenu taxonomique d'un métagénome. Le second, pauda [89], utilise une méthode originale pour comparer des séquences d'adn à des protéines.

Le serveur mg-rast [73] regroupe de nombreux outils dédiés à la métagénomique. Il fonctionne en trois étapes distinctes. Dans la première étape, mg-rast procède à une normalisation des données afin de retirer les séquences présentes en double dans l'échantillon, évitant ainsi des recherches inutiles dans les bases de données.

La seconde étape consiste alors à rechercher parmi toutes les séquences, celles codant pour des protéines. Pour cela, le logiciel blastx est utilisé sur plusieurs bases de données protéiques. En parallèle, mg-rast recherche les séquences d'arn ribosomiques potentiellement présentes dans le métagénome. Le logiciel blastn est utilisé pour cette recherche dans diverses bases d'arnr, dont rdp [60] mentionné sous-section 2.1.1. Finalement, mg-rast recherche dans l'échantillon les séquences d'adn non-nucléaire, l'adn mitochondrial et l'adn chloroplastique.

Dans la troisième et dernière étape, toutes les réponses à ces requêtes sont intégrées et permettent de calculer trois résultats. Tout d'abord, mg-rast utilise les informations d'arnr et de phylogénie des protéines trouvées pour procéder à une reconstruction de la phylogénie de l'échantillon. Ensuite, le logiciel assigne une fonction aux séquences codant pour des protéines et procède à la classification fonctionnelle de toutes ces séquences. Finalement, mg-rast utilise toutes ces informations pour déduire les possibilités de réactions enzymatiques et de flux métaboliques susceptibles de se dérouler dans le milieu étudié, et propose un début de réseau métabolique.

Mg-rast est un système largement utilisé en métagénomique fonctionnelle. Son utilisation se fait exclusivement en ligne, sur un serveur dédié à son fonctionnement ; ceci nécessite donc d'y envoyer l'intégralité des séquences à étudier. Pour de très grands métagénomes, cet envoi peut nécessiter plusieurs dizaines heures. En outre, l'utilisation massive des dérivés du logiciel blast demande de nombreuses heures de calcul pour mener à bien la recherche de similarité dans les bases de données, bases grandissant de jour en jour.

Img/m [72] offre des fonctionnalités assez proches de mg-rast, la différence majeure est qu'il n'est possible d'utiliser que les métagénomes déjà présents dans img/m. Ce logiciel

compte actuellement 2 008 métagénomes publics. Ces métagénomes ont été analysés sous de nombreux angles : détections des régions codantes, analyses d'ARNr, classification de fonctions protéiques, annotations des séquences connues, etc.

Ce logiciel propose de nombreux outils pour parcourir ces métagénomes et rechercher, par exemple, une fonction précise, un gène donné ou un organisme. Il est aussi possible d'obtenir des informations phylogénomiques sur l'ensemble de microorganismes d'un métagénome, ou seulement sur ceux sélectionnés par l'utilisateur. Différentes analyses comparatives sont alors réalisables. Par exemple, on peut rechercher les organismes communs entre plusieurs métagénomes. De même, on peut comparer les profils d'abondances de plusieurs métagénomes, procéder à des comparaisons sur les différentes fonctions exprimées par des métagénomes, etc.

Contrairement à MG-RAST, la plupart des calculs demandant beaucoup de temps sont déjà réalisés avant que l'utilisateur n'en ait besoin. Ce significatif gain de temps n'est possible que par la restriction des métagénomes utilisables. Ce dernier point est un désavantage majeur d'IMG/M : on ne peut pas travailler sur des données fraichement séquencées.

MEGAN [74] est un logiciel utilisé pour explorer le contenu taxonomique d'un échantillon. Il fonctionne en trois étapes. La première étape consiste à comparer toutes les séquences du métagénome contre les séquences référencées en base de données en utilisant BLAST. La seconde étape utilise les informations taxonomiques du NCBI (N*ational* C*enter for* B*iotechnology* I*nformation*, États-Unis) pour annoter les résultats et homogénéiser les informations des différentes bases de données utilisées précédemment. Finalement, les séquences sont assignées à un taxon particulier suivant leur similarité. Une séquence spécifique à une espèce est assignée à un taxon proche de cette espèce, tandis qu'une séquence très conservée et partagée entre plusieurs espèces sera assignée à un niveau plus haut de taxonomie. MEGAN génère un arbre phylogénétique correspondant au métagénome, mais dans lequel aucune espèce n'apparaît. Seuls les taxons et leur abondance respective sont représentés.

Ce logiciel est souvent utilisé car il peut fonctionner sur un ordinateur classique, il ne nécessite pas d'envoyer ses séquences sur un serveur distant. Ceci permet d'analyser plus rapidement les séquences, comparé à MG-RAST. Les auteurs précisent tout de même que pour de grands jeux de données, l'utilisation de BLAST et de plusieurs bases de données peut prendre énormément de temps.

PAUDA[89] est un logiciel récent dédié à la recherche de séquences métagénomiques codant pour des protéines. Classiquement, le logiciel BLASTX est utilisé pour comparer chaque séquence aux bases de données de références. L'idée principale de PAUDA est de convertir toutes les séquences codant pour des protéines, dans une base de données de références, en "pseudo ADN" (PADN).

Pour ce faire, PAUDA utilise la matrice de substitution de BLASTX. Cette matrice, BLOSUM62 [90], représente la probabilité qu'un des vingt acides aminés soit substitué par un autre acide aminé. En effet, certains acides aminés ont plus de chance d'être substitués par un autre, de par à la fois la redondance du code génétique, et les propriétés physico-chimiques de ces acides-aminés. Grâce à cette matrice, quatre groupes d'acides aminés sont définis, chaque groupe contenant les acides aminés les plus proches entre eux. Ces quatre groupes sont définis comme étant le groupe A, le groupe C, le groupe G et le groupe T.

Ainsi, toutes les séquences protéiques de la base de données de références sont converties en PADN. Les séquences du métagénome sont converties en séquences protéiques, puis ces séquences protéiques sont à leur tour converties en PADN. Un logiciel d'alignement de séquences nucléiques, BOWTIE2 [91], est utilisé pour comparer chaque séquence de PADN du métagénome contre la banque de PADN. Quand deux séquences de PADN sont identifiées comme identiques, leurs séquences protéiques sont alignées afin de calculer leur véritable similarité.

Cette nouvelle approche est 10 000 fois plus rapide que BLASTX, mais ne retrouve qu'un tiers des séquences identifiées par ce dernier. Son avantage principal est de pouvoir traiter de très gros échantillons métagénomiques, comportant des millions de séquences sur un ordinateur actuel, chose qui demande énormément de temps en utilisant BLASTX. Par contre, en terme de métagénomique fonctionnelle, PAUDA est limité aux analyses protéiques et n'apporte aucune information taxonomique.

CONCLUSION Comme pour les deux familles de métagénomique précédents, il est essentiel en métagénomique fonctionnelle d'avoir des espèces de référence bien connues. Par contre, à l'inverse de la métagénomique quantitative, des séquences proches dans le métagénome ne sont pas problématiques. En effet, des séquences proches proviennent probablement de gènes orthologues, aux fonctions similaires.

Certains logiciels de métagénomique fonctionnelle permettent de faire des analyses comparatives de métagénomes sur la base des séquences connues qu'ils partagent. Cependant, aucune des méthodes présentées plus haut ne permet de faire une analyse comparative en se basant sur **l'ensemble** des séquences contenues dans différents métagénomes. Dans la partie suivante, la métagénomique comparative est présentée, et plus particulièrement la métagénomique *de novo*.

2.1.4 Métagénomique comparative

OBJECTIFS La métagénomique comparative s'attache à comparer plusieurs métagénomes entre eux. Comparer deux métagénomes ou plus est un moyen efficace de comprendre comment les différences génomiques d'une communauté affectent les facteurs physico-chimiques d'un écosystème. Par exemple, Ishii *et al.* ont comparé différents métagénomes impliqués dans la dénitrification de plusieurs sols différents [92]. Complémentairement, cette approche informe aussi sur la manière dont l'environnement agit sur différents métagénomes [37]. Par exemple, il a été montré qu'un environnement riche en nickel favorise l'apparition de gènes de résistance à cet élément comparé à un environnement de contrôle [93]. La métagénomique comparative permet aussi de regrouper différents métagénomes sur la base de leurs contenus. Dans une étude du global ocean sampling expedition (GOS), les métagénomes ont été analysés sur la base de leurs contenus en séquences puis clusterisés. Les résultats montrent que les métagénomes provenant d'endroits proches géographiquement ou partageant des facteurs environnementaux communs tendent à se regrouper ensemble [94].

MÉTHODES La métagénomique comparative peut se baser sur différentes approches. Plusieurs études utilisent des informations de métagénomique fonctionnelle ou quantitative. Entre autres, les informations des ARN 16S sont utilisées pour comparer la composition de plusieurs métagénomes [95]. Le logiciel MG-RAST est utilisé pour des analyses taxonomiques comparatives [96], MEGAN pour des analyses de comparaison d'abondance de différents taxons [97] et IMG/M pour, par exemple, étudier les liens de parenté entre différents écosystèmes [98]. Toutes ces études se basent sur les connaissances actuelles et laissent de côté une partie des séquences [69]. Pour utiliser toutes les séquences disponibles, plusieurs approches sont utilisées.

Dans certaines études [99, 100], la proportion de cytosine (C) et de guanine (G) parmi tout l'ADN étudié, appelée taux de GC, est utilisée pour comparer les métagénomes. En effet, le taux de GC n'est pas homogène dans un génome, ni entre différents génomes [101]. L'étude de Foerstner *et al.* [99] a ainsi démontré que l'environnement a un impact considérable sur le taux de GC des organismes qui y vivent.

Raes *et al.* [102] ont développé une autre méthode pour la comparaison de métagénomes sans utiliser d'informations issues des connaissances actuelles. Ils utilisent un ensemble de

marqueurs génétiques considérés comme essentiels pour la survie des cellules et très anciens : ces marqueurs évoluent lentement et sont des membres importants de processus clé de la cellule. Les marqueurs choisis sont conservés à travers le vivant et n'apparaissent qu'une seule fois par génome. À partir de la quantité de ces marqueurs présents dans le métagénome, la taille moyenne des génomes (nommée EGS, pour *Effective Genome Size*) contenus dans l'échantillon est extrapolée. Même dans des métagénomes complexes, le nombre total de ces marqueurs est proportionnel au nombre de génomes présents. Alors, la densité de ces marqueurs (nombre de marqueurs trouvés sur le nombre total de séquences) est inversement corrélée à la taille moyenne des génomes de l'échantillon. Il est possible de comparer plusieurs métagénomes sur la base de cet EGS.

Plusieurs outils, dédiés à la génomique, s'intéressent à comparer des séquences génétiques entre elles. En particulier, BLAT [103] et UBLAST [104] sont deux outils qui servent à faire des alignements locaux entre deux jeux de séquences. Dans le cadre de la métagénomique comparative, ces deux outils peuvent servir à identifier les séquences semblables entre deux métagénomes. CRASS [105] est un outil récent qui se consacre exclusivement à la métagénomique comparative sans recours aux données de références.

MOYENS BIO-INFORMATIQUES BLAT [103] est un logiciel de génomique qui peut être utilisé en métagénomique. Il est dédié à l'alignement local et fonctionne en deux étapes. La première étape consiste à rapidement identifier les régions de deux séquences probablement homologues. La seconde étape consiste alors à analyser plus en détail ces régions, puis à produire un alignement des régions effectivement homologues. Le but de la première étape est de détecter le plus de régions homologues possibles et de réduire le nombre de séquences à analyser dans la seconde étape, afin d'éviter des calculs inutiles.

La première étape consiste à rechercher les k-mers qui sont communs entre la séquence requête et la base de données de séquences. Pour cela, BLAT indexe les k-mers non chevauchant présents dans la base de données. Les k-mers apparaissant trop souvent dans cet index sont éliminés. BLAT recherche alors dans l'index la présence de tous les k-mers de la séquence requête. Il construit une liste des résultats qui indique sur quelles parties la séquence requête et les séquences de l'index sont identiques. À la fin de ce processus, les séquences requêtes qui s'alignent peu sur une séquence de la base de données (seulement quelques k-mers ont été trouvés) sont éliminées.

La seconde étape consiste à étendre les régions autour des k-mers communs à deux séquences. Chacune de ces régions est étendue en comparant les nucléotides deux à deux. Certaines régions étendues peuvent alors fusionner. Les régions étendues qui se suivent, à la fois dans la séquence requête et dans la séquence en base de données, sont liées entre elles et forment un début d'alignement. Les espaces entre ces régions communes sont à nouveaux analysés comme autant de séquences requêtes et de séquences en base de données, en utilisant un k plus petit. Le processus est récursif et s'arrête quand les espaces sont comblés ou quand leur taille respective est inférieure à 5 nucléotides, et l'alignement final est ainsi obtenu.

La principale innovation de BLAT est l'utilisation d'un index de k-mers non chevauchants, permettant de rapidement identifier les séquences susceptibles de s'aligner correctement. Grâce à cette spécificité, BLAT est environ 500 fois plus rapide que BLAST, le logiciel de référence pour les alignements locaux. Malgré ces performances, il reste très lent d'utilisation pour aligner, et donc comparer, toutes les séquences de deux génomes. Par exemple, l'auteur estime qu'il faudrait 12 jours, en utilisant 100 processeurs, pour aligner les séquences brutes, donc non-assemblées, de deux génomes de souris. Comparer ainsi deux métagénomes n'est donc pour le moment pas réalisable en un temps raisonnable.

UBLAST [104] est aussi un logiciel de génomique mais peut être utilisé en métagénomique. Il ne recherche pas toutes les séquences pouvant s'aligner sur la requête, mais en recherche

une ou quelques-unes, suivant le choix de l'utilisateur. Comme BLAT, il se base sur l'utilisation de *k*-mers pour sélectionner les séquences les plus susceptibles de s'aligner à la requête. Pour cela, il compare tous les *k*-mers de la séquence requête avec les *k*-mers (chevauchants ou partiellement chevauchants) des séquences en base de données. La séquence partageant le plus de *k*-mers avec la requête est la première à être alignée. Pour cela, *ublast* recherche des *k*-mers plus petits s'alignant parfaitement entre les deux séquences, puis étend ces régions des deux côtés.

L'intérêt de cette méthode réside dans la sélection des meilleurs candidats avant tout autre calcul. En effet, si la requête ne s'aligne pas correctement sur les premières séquences testées, il y a très peu de chance qu'elle s'aligne bien sur des séquences partageant moins de *k*-mers. Ceci permet à UBLAST d'être environ 300 fois plus rapide que BLAST. En terme de métagénomique comparative, le défaut principal de UBLAST est de ne retourner que les premiers meilleurs résultats et non toutes les séquences semblables. De plus, étant plus lent que BLAT, il ne peut pas être utilisé sur de très gros métagénomes.

Le programme CRASS[105] permet de comparer plusieurs métagénomes sur la base des séquences communes qu'ils partagent. Pour cela, il est nécessaire, avant d'utiliser CRASS, de lancer un logiciel d'assemblage sur l'ensemble des séquences, tous métagénomes confondus. L'assemblage génère alors des contigs pouvant provenir de plusieurs métagénomes à la fois. Le fichier contenant les informations sur les contigs assemblés et l'ensemble des métagénomes à analyser servent d'entrée pour CRASS. Pour chaque contig produit par l'assembleur, CRASS compte le nombre de séquences de chaque métagénome qui ont servi à le construire. Ainsi, il détermine si certains contigs sont partagés par plusieurs échantillons. Le degré de relation entre deux métagénomes est calculé à partir de ce comptage par quatre formules de distance.

Les deux premières distances, nommée "shot" [106] et "Minimum", sont basées sur le nombre de contigs qui contiennent des séquences provenant d'un seul des métagénomes à analyser et sur le nombre de contigs qui contiennent des séquences de plusieurs des métagénomes. Ces deux distances sont des formules basées sur la présence/absence de contigs. Elles partent du principe que la présence d'un contig dans un métagénome (*i.e.* le contig contient au moins une séquence présente dans cet échantillon) est plus informative que l'abondance des séquences du métagénome dans ce contig. Cette comparaison qualitative est pertinente dans le sens où le nombre de séquences assemblées dans un contig n'est pas forcément représentatif de la communauté séquencée : différents biais peuvent arriver, tant à l'étape de prélèvement, de séquençage que d'assemblage. Mais, des informations quantitatives sont aussi fournies par CRASS, dans les deux autres formules de distance.

La troisième formule, nommée distance "Wootters" [107], utilise la quantité de séquences des deux métagénomes impliqués dans chacun des contigs. Des contigs de grande taille ont plus de chance d'attirer par hasard des séquences de métagénomes non-corrélés entre eux que des contigs de petite taille. Pour pallier cet effet, les auteurs ont créé une quatrième formule de distance.

Cette dernière formule est un mélange des concepts des trois précédentes formules. La distance "Reads" est basée sur le nombre de séquences présentes dans des contigs qui contiennent des séquences d'un seul des deux métagénomes à analyser, sur le nombre de séquences présentes dans des contigs qui contiennent des séquences de l'autre métagénome et sur le nombre de séquences présentes dans des contigs qui contiennent des séquences des deux métagénomes.

Grâce à ces quatre distances différentes, CRASS permet de comparer deux à deux des métagénomes et délivre un arbre phylogénétique, par distance, de tous les métagénomes analysés. Cette méthode est dédiée à la métagénomique et fonctionne sans recours aux données de références. Le désavantage principal de CRASS est le pré-traitement des données par assemblage. En effet, les résultats ne sont pas toujours homogènes suivant les assembleurs utilisés. De plus,

cette étape d'assemblage est très coûteuse en temps. Un métagénome contenant 3 592 984 séquences d'environ 400 pb est assemblé avec Newbler [29] en 24 heures et 50 minutes sur un processeur 8 cœurs (un seul étant utilisé en pratique) et 64 Go de RAM. Finalement, le choix de la formule de distance à utiliser en fonction de ses besoins n'est pas défini.

CONCLUSION La métagénomique comparative peut être divisée en deux parties. La première est basée sur les données déjà référencées en base de données et utilise des logiciels d'annotations, de phylogénie, de classification, etc. Le but est d'identifier des références connues dans les différents métagénomes et de comparer les échantillons sur la base de ces références. Ces méthodes n'utilisent pas l'ensemble des données présentes dans les métagénomes, car une partie parfois très importante des séquences est inconnue.

La seconde approche, utilisant la totalité des données présentes dans l'échantillon, sera désignée dans cette thèse par **métagénomique comparative *de novo***. Certaines approches génomiques, comme BLAT ou UBLAST, peuvent être utilisées pour comparer plusieurs métagénomes sur la base du nombre de séquences qu'ils partagent. Ces logiciels n'étant pas développés pour une approche métagénomique, ils ne permettent pas de comparer directement plusieurs métagénomes. La méthode utilisée par CRASS, consistant à passer par une phase d'assemblage croisé des différents métagénomes, rentre dans le cadre de la métagénomique comparative *de novo*. La nécessité de passer par un logiciel d'assemblage externe peut être problématique quand on souhaite comparer plusieurs dizaines de métagénomes, les assembleurs classiques n'étant pas développés pour utiliser autant de données. Dans plusieurs grands projets métagénomiques (voir sous-section 2.2), la comparaison de métagénomes est intensive. Le projet Metasoil (voir sous-section 2.2.2) a par exemple généré et comparé 13 métagénomes de terre, le projet GOS (*Global Ocean Sampling*, voir sous-section 2.2.3) a analysé 41 métagénomes, et le projet Tara Oceans (voir sous-section 2.2.4) devrait aboutir à l'analyse de plus de 2000 métagénomes.

La métagénomique comparative *de novo* ne dispose donc pas d'outils permettant d'analyser de nombreux métagénomes directement, sans recours à des données de références, ni à d'autres logiciels en amont. Cette thèse vise à développer un tel outil, capable de comparer *de novo* un grand nombre de métagénomes (plusieurs centaines par exemple), contenant chacun un très grand nombre de séquences (plusieurs centaines de millions pour les technologies de séquençage actuelles) et sans utiliser de données connues. Comme les approches UBLAST ou BLAT, la première étape de cet outil est d'indexer un très grand nombre de k-mers, pour ensuite procéder à des comparaisons entre ces k-mers. Pour cela, plusieurs structures de données informatiques peuvent être utilisées et seront détaillées sous-section 2.3.

2.2 QUELQUES PROJETS MÉTAGÉNOMIQUE DE GRANDE ENVERGURE

Ces quinze dernières années, de nombreux projets métagénomiques ont vu le jour. Parmi ces projets, quatre ont eu recours à un séquençage massif de dizaines, voire de centaines de métagénomes. Ces études sont basées sur des milieux très différents. Le projet MetaHit étudie le microbiome intestinal humain. L'étude Metasoil s'intéresse à des échantillons de sols "sains", qui n'ont pas été utilisés par l'homme depuis plus de 150 ans. Finalement, Global Ocean Sampling et Tara Oceans se concentrent sur les microorganismes vivant dans différents océans du globe.

2.2.1 MetaHIT

MetaHIT (METAgenomics of the Human Intestinal Tract) est un projet collaboratif financé par la commission européenne et rassemblant 15 instituts de 8 pays différents [108]. Il a pour but d'utiliser les NGS pour explorer le microbiome humain et comprendre les relations entre ces microorganismes et la santé humaine. La quantité de microorganismes vivant dans un corps humain est estimée à 10 000 milliards de cellules, pour un poids d'environ 2 kg [109]. De nombreuses maladies ont une origine microbienne, comme les maladies touchant le système digestif. Certaines maladies chroniques ont récemment été liées à un changement inhabituel du microbiome [110, 111]. De même, certaines maladies apparemment psychologiques, comme des formes d'autisme, semblent être liées à certains microorganismes vivant dans l'intestin [112].

2.2.1.1 *Enjeux*

Le projet MetaHIT a choisi d'étudier en détail deux maladies en augmentation dans nos sociétés modernes : l'obésité et les maladies inflammatoires chroniques intestinales (MICI), particulièrement les deux principales formes que sont la maladie de Crohn et la colite ulcéreuse. Ces maladies sont des inflammations de l'intestin à caractère chronique.

Le projet MetaHIT a trois objectifs. Le premier est d'établir un catalogue le plus complet possible des gènes et génomes microbiens présents dans l'intestin humain. Le second objectif est de développer des outils pour déterminer quels gènes ou génomes du catalogue de référence sont présents chez différents individus et à quelle fréquence. Le troisième objectif est d'étudier près de 600 personnes, saines ou atteintes des pathologies recherchées, pour déterminer quels gènes et génomes ces personnes portent, ceci afin d'établir des associations entre le microbiome intestinal et la santé de l'individu. En terme de métagénomique, cette étude utilise aussi bien des techniques de métagénomique fonctionnelle, quantitative, ciblée ou comparative. C'est surtout cette dernière partie, comparative, qui va être étudiée ici.

2.2.1.2 *Résultats*

À l'heure actuelle, un des principaux résultats est la création d'un catalogue regroupant tous les gènes trouvés dans les différents échantillons. Afin d'éliminer les doublons, toutes les séquences ont été comparées deux à deux à l'aide du logiciel BLAT. Ce catalogue indique que le microbiome intestinal contient plus de 3,3 millions de gènes différents, soit environ 150 fois plus que le nombre de gènes portés par l'ADN humain [69]. Les séquences de tous les métagénomes ont été alignées sur ce catalogue. Pour qu'un gène du catalogue soit trouvé dans un métagénome, il doit y avoir au moins deux séquences du métagénome qui s'alignent sur ce gène. De ces alignements, il a été déduit que chaque individu porte $536\,112 \pm 12\,167$ gènes du catalogue, indiquant qu'une grande partie des 3,3 millions de gènes du catalogue doivent être partagés entre plusieurs individus. Contradictoirement, 2 375 655 gènes du catalogue ne sont présents que dans moins de 20% des individus, et 294 110 sont retrouvés dans au moins 50% des individus. Les échantillons partagent entre eux $204\,056 \pm 3\,603$ gènes, indiquant qu'environ 38% des gènes d'un individu sont partagés. De plus, il a été remarqué que les échantillons de personnes souffrant de MICI contiennent, en moyenne, 25% moins de gènes que les échantillons de patients non atteints de cette maladie.

À l'aide de ce catalogue, le consortium MetaHIT a été capable de déterminer l'existence de trois entérotypes [113]. Un entérotype est défini comme étant un groupe spécifique de bactéries du même genre. Le premier entérotype est caractérisé par un grand nombre de *Bactéroides*, le second par une faible quantité de *Bactéroides* mais un haut niveau de *Prevotella* et le dernier, par une forte présence de *Ruminococcus*. Afin de définir ces entérotypes, les

séquences des métagénomes ont été classifiées en utilisant la base de données RDP [60] et son outil de classification [76]. Apparemment, ces entérotypes ne sont pas corrélés avec la provenance géographique des individus, ni avec leur âge ou leur sexe. Mais, un lien existerait entre le régime alimentaire et la définition de ces entérotypes [114].

Un ensemble contenant 155 espèces bactériennes, dont au moins 1% de leur génome est couvert par au moins un des individus, a été défini. L'abondance de chacun de ces génomes a été mesurée dans 39 métagénomes. L'analyse en composante principale (ACP) de ces abondances sépare clairement les individus sains des individus porteurs de la maladie de Crohn et de ceux victimes de colite ulcéreuse.

Dans le futur, le consortium espère pouvoir caractériser les maladies intestinales par certains microorganismes clés, en faisant un lien direct entre la présence ou l'absence de ces bactéries et les maladies. Ces associations permettraient de développer de nouveaux outils pour diagnostiquer et enrayer les troubles étudiés. De plus, outre une meilleure connaissance de l'humain et de son microbiome intestinal, le projet MetaHIT vise aussi à étudier les effets sur la flore intestinale de différents facteurs, comme le régime alimentaire, l'âge, ou encore la réaction à certains médicaments.

2.2.1.3 *Conclusion*

Le projet MetaHIT est un mélange de métagénomique ciblée, fonctionnelle, quantitative et comparative. Les analyses comparatives sont réalisées sur la bases des fonctions partagées, des gènes présents ou des abondances de certains microorganismes dans les différents métagénomes. Il semble qu'aucune analyse comparative *de novo* n'ait été publiée pour le moment sur ce projet. Comparer l'ensemble des séquences entre les différents métagénomes peut conduire à identifier certaines séquences systématiquement associées dans une condition particulière.

2.2.2 MetaSoil

Les propriétés physico-chimiques des sols sont des paramètres essentiels pour l'agriculture. Un sol trop acide ou trop poreux peut rendre impossible certaines cultures. Outre ces aspects, la population microbienne conditionne le bon développement de certaines plantes [115]. Le projet Metasoil a pour but d'étudier la composition microbienne d'un sol de référence, le *Park Grass Experiment* de Rothamsted, Angleterre. La comparaison des différents métagénomes prélevés dans ce sol vise à mieux comprendre l'adaptation des microorganismes aux changements subis (saisons, profondeurs, etc) et à étudier le rôle de certains microorganismes dans différents écosystèmes [116].

PARK GRASS EXPERIMENT Le parc choisi pour l'échantillonnage est très particulier. Le Park Grass Experiment est la plus ancienne étude continue de sol. Le terrain sur lequel elle se déroule est internationalement connu comme étant le lieu de nombreuses études depuis plus de 150 ans : ce terrain de 28 000 m² est le lieu d'expériences sur la biodiversité et la sélection naturelle depuis 1856. Il a donc été préservé de toutes techniques modernes d'agriculture, incluant engrais et désherbants, et en fait un excellent terrain neutre [117].

2.2.2.1 *Enjeux*

Un des enjeux de Metasoil est de comparer l'influence sur le séquençage de trois paramètres : la profondeur sous la terre de l'échantillonnage, les saisons et plusieurs techniques d'extraction et de purification d'ADN [118]. Pour cela, la comparaison de métagénomes s'effectue en utilisant la métagénomique quantitative et fonctionnelle. Cela consiste à comparer les métagénomes venant de différentes conditions sur la base de leur composition en espèces

et des informations fonctionnelles [116]. Les auteurs estiment que des comparaisons globales entre différents métagénomes peuvent aider à définir les spécificités d'un écosystème donné, comme les réponses qu'il exprime suite à un changement climatique ou à l'apparition d'un pathogène, ou encore des capacités de biodégradation d'un polluant particulier. Un autre but de l'étude Metasoil est de constituer un catalogue de ressources génétiques et métagénomiques d'un sol de référence internationalement reconnu. À partir de cette base de connaissances sur les gènes, fonctions et génomes contenus dans un sol "sain", il sera possible de faire des corrélations entre des perturbations d'un environnement et les changements observés du microbiome correspondant.

2.2.2.2 *Résultats*

L'annotation des séquences a été conduite à l'aide MG-RAST [73]. L'analyse fonctionnelle de ces séquences a permis d'identifier 835 fonctions présentes dans au moins un des échantillons étudiés. Ces fonctions ont servi de référence pour différentes études. Toujours à l'aide de MG-RAST, une analyse quantitative des ces références a été menée sur les 13 métagénomes de Rothamsted, deux métagénomes provenant d'autres sols et un métagénome d'eau de mer. Les échantillons ont ensuite été clusterisés sur la base du nombre de fonctions de références qu'ils partagent les uns avec les autres. Les groupes issus de cette clusterisation montrent une bonne corrélation avec la technique d'extraction d'ADN utilisée, indiquant d'importants biais lors de cette étape [119]. Par contre, aucune corrélation au niveau fonctionnel n'a pu être établie entre la profondeur des échantillons ou le rythme des saisons [120].

Dans une des publications de Metasoil, les auteurs ont conclu que les limitations et les biais des différentes techniques d'extraction d'ADN doivent être pris en compte au sein d'un projet incluant plusieurs séquençages. Toutefois, en comparant des métagénomes provenant d'environnements très éloignés, comme de l'eau de mer et du sol, ces biais d'extraction sont suffisamment ténues pour permettre de déterminer les différences entre les communautés microbiennes présentes dans les échantillons. Ainsi, ils expliquent que l'on peut mener des analyses comparatives sur, par exemple, les fonctions exprimées ou les distributions d'espèces entre des métagénomes éloignés, même si les échantillons proviennent de projets différents [116]. De même, ils insistent sur l'importance de mener des analyses de métagénomique comparative entre différents échantillons et différents milieux, cela afin d'extraire plus d'informations qu'en se concentrant sur les fonctions exprimées, ou les espèces présentes, dans un seul métagénome [121].

2.2.2.3 *Conclusion*

La caractérisation du métagénome d'un sol "sain", ou "neutre", peut servir de base à de futures analyses. La vision globale du microbiome vivant à l'état naturel dans un sol peut être utilisée pour des questions spécifiques, par exemple sur les dommages à court et long terme d'un polluant, d'un engrais ou d'une technique d'agriculture donnée. De même, cette référence de sol peut permettre des analyses entre différents environnements pour caractériser les ressemblances ou divergences entre un sol "classique" et un autre milieu. Une des limites majeures de cette étude est le manque de références dans les bases de données. Ainsi, seul 1% des séquences annotées correspond à un organisme connu et séquencé. De même, toutes les séquences générées ne sont pas utilisées : seules $34{,}5(\pm 3{,}3)\%$ des séquences sont effectivement annotées en utilisant MG-RAST. Dans le cadre du présent doctorat, une analyse de métagénomique comparative *de novo*, donc utilisant toutes les séquences, a été menée sur l'ensemble des métagénomes de Metasoil. Les résultats de cette étude concordent avec ceux décrits plus haut et sont présentés sous-section 4.3.3.

2.2.3 Global oceans sampling

L'expédition global ocean sampling (GOS) a été menée par le laboratoire JCVI (*J. Craig Venter Institute*). Durant cette expédition, des échantillons d'eau de mer ont été prélevés dans 41 lieux différents, entre l'océan Atlantique nord, le canal de Panama et le sud de l'océan Pacifique. Chacun des prélèvements a été séquencé et l'étude a fourni 44 métagénomes représentatifs d'une partie des océans du globe. Cette expédition a conduit à enrichir les connaissances sur le milieu marin, tant sur le plan des espèces qui y vivent que sur les fonctions ou protéines exprimées par ces organismes [100]. Dans l'une des études associées au projet [94], les auteurs ont procédé à une comparaison brute des métagénomes sur la seule base de leur contenu en séquences.

2.2.3.1 *Enjeux*

L'enjeu principal de GOS était d'étudier la diversité génétique des communautés microbiennes vivant dans l'eau de mer. Cette étude a porté sur la composition en espèces des différents métagénomes, sur l'abondance relative des ces espèces entre les différents échantillons [94], mais aussi sur les protéines synthétisées par les différentes communautés microbiennes [100].

2.2.3.2 *Résultats*

Le premier résultat fut l'obtention de 44 métagénomes aquatiques provenant de 41 lieux différents. La classification et la phylogénie des séquences contenues dans ces métagénomes ont été réalisées avec RDP [60] et son outil de classification [76], MUSCLE [59], BLAST [22] et d'autres outils. Les auteurs ont ensuite procédé à l'assemblage de chaque métagénome en utilisant une version modifiée de celera [28], dans lequel seul 9% des séquences étaient dans des scaffolds de plus de 10 kpb. Plus de la moitié du total des séquences (53%) n'a pas du tout été assemblée. À l'époque de la publication, en 2007, ils ont utilisé 334 génomes complets et 250 génomes non finis comme références. En utilisant BLAST de manière très permissive (55% d'identité de séquence), 70% des séquences de GOS s'alignaient sur un ou plusieurs génomes de référence. Avec des paramètres plus restrictifs (les séquences doivent s'aligner sur presque toute leur longueur et sans trous de grande taille), environ 30% des séquences s'alignaient sur au moins une référence.

Le résultat le plus intéressant en terme de métagénomique comparative *de novo* est la création d'une méthodologie pour comparer la totalité des séquences de deux métagénomes. Plutôt que d'utiliser des approches traditionnelles comme l'analyse de ARNr 16S, les auteurs ont développé une méthode pour déduire la similarité génétique entre deux échantillons en utilisant toutes les séquences disponibles. Cette similarité revient à estimer quelle fraction des séquences d'un métagénome peut être considérée comme présente dans un autre métagénome. En calculant les similarités pour chaque paire d'échantillons, on obtient une matrice de similarité entre tous les échantillons.

La première étape de cette comparaison est la construction d'une base de données de séquences chevauchantes. Pour cela, toutes les séquences de tous les métagénomes ont été comparées les unes contre les autres. La comparaison a été effectuée avec un composant modifié de Celera qui détecte les chevauchements parfaits d'au moins 14 nucléotides entre deux séquences. À partir de ce point d'ancrage, l'alignement de séquences est effectué en comparant les nucléotides à gauche et à droite de l'ancre. Tant que l'alignement conserve une identité de plus de 65%, la zone chevauchante augmente. Les séquences se chevauchant de cette manière sur plus de 40 nucléotides sont conservées. La base de données finale contenait 1,2 milliards de chevauchements.

Le calcul de similarité entre deux échantillons i et j utilise cette base de données de chevauchement. Pour une séquence s de i, soit $n_{s,i}$ le nombre de séquences de i qui chevauchent s avec une identité de séquence minimale, par exemple 90%, et $n_{s,j}$ le nombre de séquences de j qui chevauchent s avec cette même identité de séquence. On calcule alors la fraction des séquences chevauchant s qui viennent de i ainsi : $f_{s,i} = n_{s,i}/(n_{s,i} + n_{s,j})$. De même, $f_{s,j} = n_{s,j}/(n_{s,j} + n_{s,i})$.

On calcule $t_{i,i}$ qui est la somme des $f_{s,i}$ pour toutes les séquences s de i et $t_{j,j}$ pour les séquences s de j (voir équation 2).

Alors, $t_{i,j}$ est la somme des fractions des séquences chevauchant s qui viennent de j. De même pour $t_{j,i}$ (voir équation 3).

On calcule alors la similarité entre i et j par l'équation 4.

$$t_{i,i} = \sum_s f_{s,i} \tag{2}$$

$$t_{i,j} = \sum_s 1 - f_{s,i} \tag{3}$$

$$S_{i,j} = S_{j,i} = \frac{2 \times r_{i,j}}{1 + r_{i,j}} \tag{4}$$
$$avec\ r_{i,j} = r_{j,i} = \frac{0{,}5 \times (t_{i,j} + t_{j,i})}{\sqrt{t_{i,i} \times t_{j,j}}}$$

Le résultat de la similarité entre i et j est compris entre 0 et 1. Un score de 0 indique qu'aucune séquence de i et de j ne se chevauche. Un score de 1 indique qu'une séquence de i et une séquence de j ont autant de chance de se chevaucher que deux séquences de i ou deux séquences de j.

La figure 7 a été obtenue pour une identité de séquence d'au moins 90%. 38% des séquences de tous les métagénomes ont participé à ce résultat. On remarque clairement la présence de groupes d'échantillons. Les échantillons sont la plupart du temps par groupe reflétant leur origine géographique. Les échantillons venant d'eau hyper-saline (33) et d'eau douce (20) sont exclus des autres échantillons venant d'eau de mer. L'échantillon 25 est aussi mal regroupé avec les autres échantillons. La raison proposée par les auteurs est que cet échantillon n'a pas été filtré de la même manière que les autres ; un filtre plus large, donc laissant passer des microorganismes plus gros que lors des autres prélèvements, a été utilisé.

2.2.3.3 *Conclusion*

Cette étude de métagénomique comparative *de novo* a permis de montrer que la seule composition en séquences des métagénomes permet de retrouver de l'information, en l'occurrence ici, de regrouper les échantillons de par leur provenance géographique. Toutes les séquences ne sont pas utilisées, mais il n'y a pas de pré-sélection des séquences intervenant dans le calcul de similarité. L'utilisation massive du module d'alignement modifié de celera a permis d'identifier les séquences chevauchantes, mais aucune notion de temps d'exécution ou de ressource mémoire ne sont mentionnées dans l'article. De plus, les séquences effectivement similaires entre deux métagénomes ne sont pas directement récupérables pour une analyse plus en détail de leur fonction ou organisme d'appartenance. Enfin, le score de similarité entre deux échantillons peut être trompeur. Prenons i et j qui contiennent chacun trois séquences différentes, et seule une de ces six séquences est commune aux deux jeux ($i = a,b,c$ et $j = a,d,e$). Le score de similarité sera de $S_{i,j} = \frac{1}{3}$. Prenons maintenant deux échantillons k et l tel que k est composé d'une seule séquence répétée cinq fois et l composée de cinq séquences différentes. Seule une séquence de l est présente dans k ($k = a,a,a,a,a$ et $l = a,b,c,d,e$). Le score de similarité sera aussi de $S_{k,l} = \frac{1}{3}$. Or, les deux exemples sont radicalement différents. Ce score ne permet pas

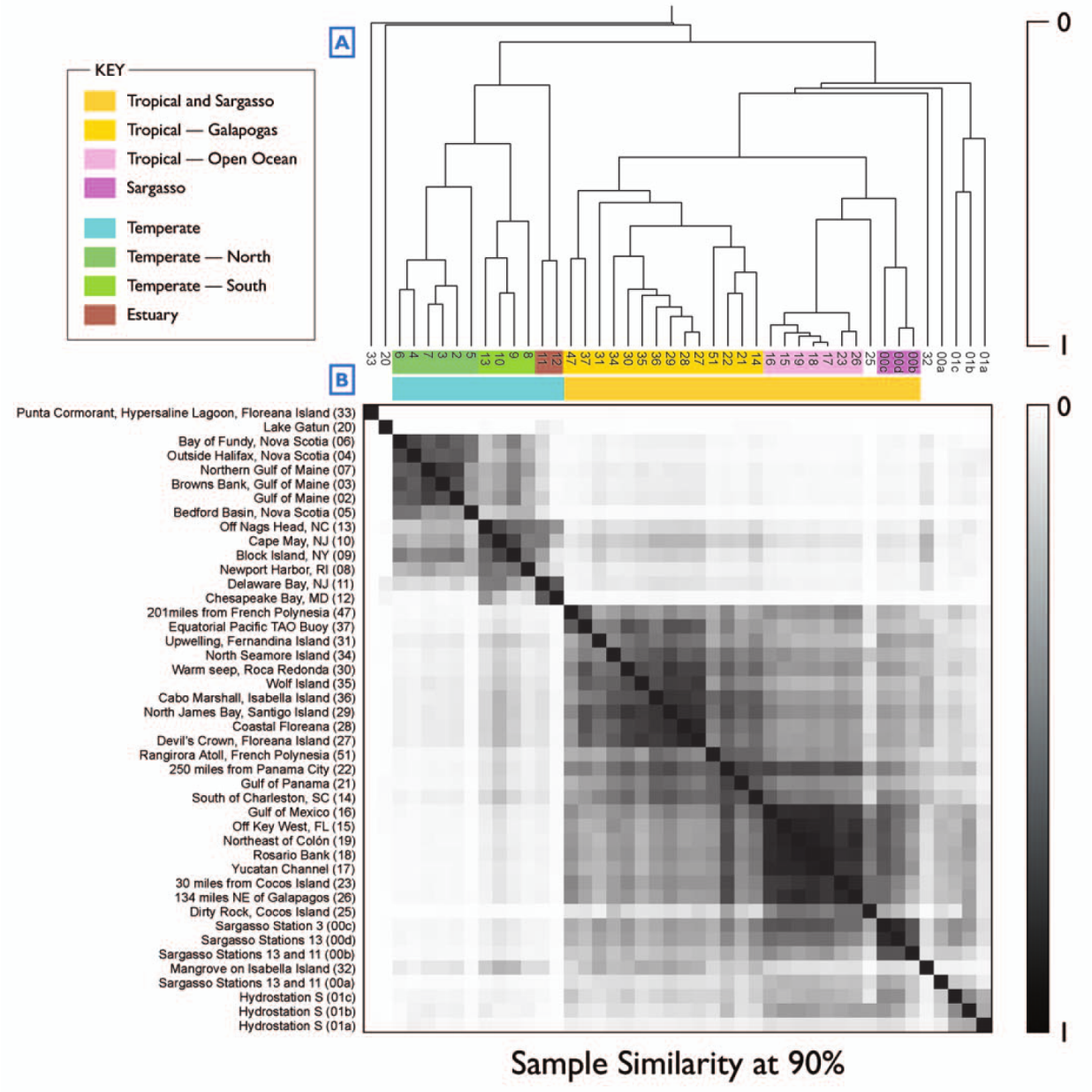


FIGURE 7: Résultat du calcul de similarité entre tous les échantillons de GOS avec une similarité de séquences d'au moins 90%. Un score de 0 indique que les deux échantillons concernés ne partagent pas de séquences en commun. Un score de 1 indique qu'une séquence du premier échantillon et une séquence du second échantillon ont autant de chance de se chevaucher que deux séquences du premier échantillon ou deux séquences du second échantillon . Figure issue de Rusch *et al.*[94].
La plupart des échantillons venant des tropiques ou de la mer des Sargasses sont regroupés ensemble après clusterisation hiérarchique. De même, les échantillons provenant de la côte EST des ÉTATS-UNIS sont regroupés ensemble. À un degré plus fin, on remarque que les échantillons entre le NORD et le sud de la côte EST des ÉTAT-UNIS forment deux groupes distincts. De même, les deux échantillons d'eau d'estuaire sont regroupés ensemble. De telles subdivisons sont aussi visibles dans les eaux tropicales.

d'associer une distance de similarité par échantillon, indiquant combien de séquences d'un échantillon sont retrouvées dans le second, et inversement. Pour i et j, on aurait bien $\frac{1}{3}$ de i dans j et $\frac{1}{3}$ de j dans i, mais pour k et l, on aurait 100% de k dans l et 20% de l dans k.

Dans la sous-section 4.3.4, les métagénomes de GOS sont analysés avec la méthode développée durant ce doctorat. Les résultats concordent globalement avec ceux de la publication d'origine, bien que le score de similarité développé soit très différent de celui utilisé ici.

2.2.4 Tara oceans

Tara oceans est une expédition scientifique visant à analyser, décrire et étudier les microorganismes marins. La goélette Tara (voir figure 8) emmène à son bord, pendant trois ans, de nombreux scientifiques venant de disciplines très variées. Le consortium Tara Oceans réunit plus d'une centaine de scientifiques venant de différents domaines comme l'océanographie, l'écologie microbienne, la génomique, la biologie cellulaire, la biologie moléculaire, la biologie des systèmes, la taxonomie, la modélisation d'écosystèmes ou la bio-informatique. Le programme d'échantillonnage comprend des relevés optiques et génétiques de virus, bactéries, archées, protistes, et métazoaires mais aussi des conditions physico-chimiques dans lesquels évoluent ces organismes. L'étude à l'échelle du globe sur la morphologie, la génétique et la bio-diversité fonctionnelle des microorganismes planctoniques, tout cela mis en relation avec les changements physico-chimique des océans, devient critique pour comprendre et gérer nos océans [122].

FIGURE 8: Goélette de l'expédition Tara Oceans.
(Crédit photo : D. Sauveur)

2.2.4.1 *Enjeux*

L'enjeu premier du projet Tara Oceans est d'analyser et comparer les différentes populations de phages, virus, bactéries, protistes et métazoaires présents dans les océans. De plus, le projet vise à étudier l'effet des variations environnementales sur ces populations [123]. Un second objectif est l'étude de l'impact des écosystèmes planctoniques sur la Terre. Cela passe par une analyse de la composition moléculaire des espèces présentes dans les océans, mais aussi par des analyses de diversité et d'évolution des espèces. Les organismes planctoniques sont largement méconnus [5], mais recèlent des bio-activités susceptibles d'aider l'Homme dans différents domaines comme la pharmacologie, l'industrie agro-alimentaire ou la création d'énergie.

2.2.4.2 *Moyens*

La goélette Tara a prélevé de l'eau en 155 endroits différents en mer [123]. Ces arrêts sont nommés "stations". Durant chaque station, de nombreux prélèvements ont été réalisés, entre autre la collecte de plancton à deux profondeurs distinctes : une en surface et la seconde à la profondeur où la chlorophylle est maximale (DCM pour *Deep Chlorophyll Maximum*). Ce DCM représente la couche dans la colonne d'eau où le phytoplancton est le plus présent. Ces deux prélèvements conduisent chacun à plusieurs échantillons. L'eau de mer est filtrée à différentes tailles pour cibler plusieurs types de microorganismes, allant des virus jusqu'aux petits métazoaires. A chaque station, l'équipage utilisait des images satellite en temps réel et de nombreux relevés physico-chimiques pour sélectionner les masses d'eau les plus intéressantes.

Les données sont analysées tant d'un point de vue génomique que physique. Les études génomiques permettent d'enrichir les modèles fonctionnels relatifs à la biodiversité et à l'évolution des organismes planctoniques, et les études physiques raffinent les modèles physiques et dynamiques de l'évolution biogéographique de ces organismes. Les NGS permettent de séquencer une part importante de l'ADN des échantillons et donnent accès à la diversité des organismes les plus représentés dans le prélèvement biologique. Plusieurs filtres sont utilisés pour séparer les organismes de différentes tailles. Ainsi, on a un aperçu global des organismes composant un échantillon, des virus aux larves de poissons. L'utilisation massive des NGS permet aussi d'obtenir une bonne couverture des ADNr exprimés dans le milieu prélevé. Cette couverture est nécessaire pour l'identification et la quantification des principaux gènes transcrits dans la population au moment de l'échantillonnage.

Une telle quantité d'informations a nécessité le développement de pipelines combinant des outils d'acquisition de données automatique et semi-automatique. De même, de nouveaux outils bio-informatiques et des outils de standardisation et d'organisation des données ont été créés.

2.2.4.3 *Résultats attendus*

Les jeux de données générés par l'expédition Tara (plus de 2000 échantillons juste pour la métagénomique), sont un résultat en soi. Ces données forment le plus grand jeu de données d'organismes marins. De nombreuses années vont être nécessaires pour procéder à leur analyse en détail.

Ces données vont permettre d'améliorer notre connaissance des principaux écosystèmes marins. De plus, ces informations vont nous renseigner sur l'évolution de la vie dans les océans. Il sera alors possible de mieux comprendre et prédire des phénomènes macroscopiques ou globaux, comme l'évolution des stocks de poissons ou l'impact du réchauffement climatique sur les organismes marins.

Le projet tara se veut comme la systématisation du projet Global Oceans Sampling. Le projet GOS avait initié cette recherche en se focalisant sur les bactéries de surface des océans ; Tara souhaite aller plus loin en étudiant surtout les micro-eucaryotes de tous les océans du globe, à différentes profondeurs et en différenciant les organismes étudiés sur la base de leur taille. De plus, de nombreux paramètres physico-chimiques et relevés satellites sont assurés à chaque station, afin d'étudier les corrélations entre certains événements biologiques et des facteurs environnementaux [122].

2.2.4.4 *Conclusion*

Tara est la première étude, à l'échelle de la planète, qui cherche à lier biogéographie, écologie marine et génétique. Au moment de la rédaction de cette thèse, la goélette est encore en expédition, au pôle Nord. Si la campagne d'échantillonnage est presque finie, le projet n'en est qu'à son début. Ce doctorat s'inscrit dans le projet Tara Oceans, sur la partie métagénomique.

Les premiers échantillons séquencés ont confirmé que très peu de séquences d'organismes marins sont actuellement référencées en base de données. Ce milieu est très mal connu et les analyses basées sur les données connues ne sont pas suffisantes. Il est nécessaire d'analyser les jeux de données métagénomiques d'un point de vue *de novo*. L'approche développée dans la suite consiste à comparer plusieurs échantillons métagénomiques entre eux. Cette comparaison est basée sur les séquences proches que deux jeux partagent, permettant de repérer, par exemple, des séquences ubiquistes, c'est à dire présentes dans plusieurs milieux très différents. Cette comparaison permet de dériver un score de similarité entre les échantillons, score qui permet alors de regrouper les échantillons les plus proches d'un point de vue composition en séquences. Certains résultats de notre méthodologie sur Tara Oceans sont présentés sous-section 4.3.5.

Conclusion sur la métagénomique

Dans cette thèse, nous proposons une classification du paysage métagénomique, discipline qui englobe plusieurs méthodologies. Cet effort de clarification est relatif, des études peuvent nécessiter des méthodes qui ne rentrent pas dans les quatre familles proposées.

La plupart des analyses métagénomiques visent à étudier quantitativement ou fonctionnellement les organismes présents dans un échantillon. Mais ces analyses se basent exclusivement sur les données connues et référencées. Dès lors, dans certain milieux mal connus comme l'eau de mer [5], seule une fraction des séquences, parfois très minime, est utilisée. De plus en plus de gros projets voient le jour. Ces projets profitent de l'explosion des NGS qui permettent d'obtenir toujours plus de séquences à des prix en constante diminution. Ainsi, une quantité de données sans précédent dans l'histoire de la biologie est actuellement générée. Si la plupart de ces données n'est pas exploitable pour des comparaisons avec les données connues, il est tout de même possible d'en extraire de l'information, par exemple en dérivant un score de similarité entre plusieurs échantillons pris deux à deux [94].

Dans le cadre du projet Tara Oceans, on estime qu'il y aura plus de 2000 échantillons à analyser, pour un total d'un pétaoctet de données. Faire toutes les comparaisons deux à deux demande une implémentation spécifique pour être réalisable en un temps raisonnable. Ce doctorat vise donc à concevoir une méthode permettant ces comparaisons. Comme pour la plupart des logiciels d'alignement ou de comparaison de séquences, il est essentiel d'indexer les séquences. L'indexation consiste à organiser des données de manière à pouvoir rapidement y accéder. Un index autorise différentes requêtes, comme trier les données indexées, rechercher la présence ou l'absence d'un élément donné dans l'index, rechercher toutes les occurrences d'un élément dans l'index, etc. Dans notre cas, nous utilisons l'index pour tester la présence ou l'absence de séquences dans les données indexées. Plusieurs structures d'indexation de données existent en bio-informatique. Dans la dernière partie de cet état de l'art, ces structures sont présentées et mises en relation avec notre problématique de métagénomique comparative *de novo*, qui nécessite d'indexer plusieurs centaines de millions de séquences d'ADN. Nous avons donc besoin d'une structure d'indexation légère en mémoire et très rapide, tant à la phase d'indexation des données qu'à la phase de recherche.

2.3 DIFFÉRENTES STRUCTURES DE DONNÉES

Dans l'approche que nous souhaitons mettre en place, nous voulons rechercher si les séquences d'un jeu de données sont présentes dans un second jeu. Les jeux de données sont obtenus après un séquençage métagénomique : les séquences sont donc des morceaux aléatoires de la totalité de l'ADN présent dans l'échantillon. Outre les éventuelles erreurs de séquençage, rechercher si une séquence d'un jeu apparait à l'identique dans un second jeu n'a que peu d'intérêt dans notre cas. On souhaite identifier les séquences d'un jeu qui sont en partie identiques à des séquences d'un second jeu, c'est-à-dire comportant des sous-parties communes.

Pour cela, une technique largement utilisée consiste à rechercher des k-mers (des sous-séquences de taille prédéfinie, voir figure 5) communs entre une séquence requête d'un premier jeu et toutes les séquences d'un second jeu. Sans indexer le second jeu de données, une recherche dans ce jeu nécessite de lire tout les k-mers de toutes les séquences tant que les critères de recherche n'ont pas été satisfaits. L'indexation nous sert à rapidement identifier, donc sans lire l'ensemble des données, si un k-mer est présent ou non dans l'index.

Ainsi, notre approche consiste à :

A. indexer les séquences d'un grand jeu de données métagénomiques ;

B. pour chaque séquence d'un second jeu de donnée, rechercher si les k-mers de cette séquence sont présents dans l'index.

Il existe plusieurs structures de données capables d'effectuer un test de présence/absence d'un élément dans un ensemble. Les structures présentées ci-dessous sont couramment employées en bio-informatique pour indexer des séquences puis parcourir cet index à la recherche d'une séquence ou sous-séquence particulière. L'arbre des suffixes [124] est une structure qui indexe sous forme d'arbre tous les suffixes d'un mot. Cet ensemble ainsi indexé peut être interrogé en temps linéaire pour y rechercher un motif particulier, par exemple dans notre cas, une sous-séquence. Le tableau des suffixes [125] a les mêmes caractéristiques de recherche mais utilise une quantité de mémoire plus faible. Le FM-index [126] a plusieurs points communs avec le tableau des suffixes. Un de ses principaux avantages est d'être compressé, et donc plus économique en mémoire. Une structure largement utilisée en informatique pour indexer des données est la table de hachage. Dans cette structure, l'élément à indexer sert d'indice dans un tableau. Cet indice pointe donc vers une unique entrée du tableau et on peut stocker une donnée à cette entrée. Dans le cas de test de présence/absence, on stocke la séquence elle même. La recherche d'un élément dans une table de hachage est souvent considérée comme se faisant en temps constant. Une autre structure de données, le filtre de Bloom [127], a été récemment utilisée en bio-informatique. Cette structure peut être apparentée à une version simplifiée, et plus légère en mémoire, de la table de hachage.

2.3.1 Arbre des suffixes

Un arbre des suffixes [124] est une structure de données arborescente utilisée pour indexer une séquence. Une fois cette séquence indexée, il est possible d'y mener un grand nombre d'opérations complexes comme tester si une sous-séquence (ou motif) apparait dans la séquence indexée, trouver toute les occurrences d'un motif, etc. Dans notre cas, cette séquence est la succession de toutes les séquences d'ADN d'un échantillon métagénomique, chaque séquence étant séparée des autres par un caractère unique. Cette succession est par la suite vue comme étant une seule séquence S de longueur n.

L'arbre des suffixes d'une séquence S, composée de n caractères, est enraciné et contient exactement n feuilles, numérotées de 0 à $n-1$. Chaque branche de l'arbre est étiquetée par une

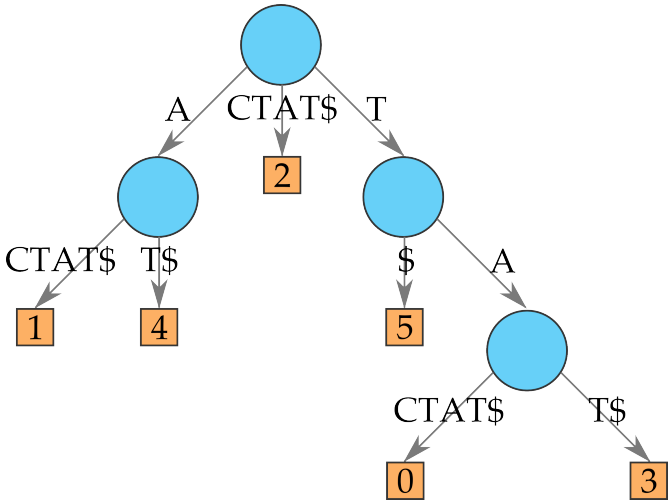


FIGURE 9: Arbre des suffixes pour le mot TACTAT$.

sous-séquence non vide de S. Chaque nœud de l'arbre, à l'exception de la racine, comporte au moins deux fils. Les fils d'un nœud commencent chacun par un caractère différent.

Pour chaque feuille i d'un arbre des suffixes, la concaténation des étiquettes portées par les branches depuis la racine vers i est le suffixe de S commençant à la position i : $S[i..n-1]$. Pour que cette définition soit vérifiée, aucun suffixe de S ne doit être un préfixe d'un autre suffixe de S ; dans un tel cas, il n'existe aucun chemin de la racine vers ce premier suffixe. Pour cela, un caractère n'apparaissant pas dans la séquence à indexer est ajouté à la fin de S. Dans ce document, ce caractère est "$".

La figure 9 représente l'arbre des suffixes de $S =$ TACTAT$. Le chemin de la racine à la feuille 0 est la séquence S. La chemin de la racine à l'étiquette 4 est la sous-séquence $S[4..n]$: AT$.

En 1995, Ukkonen propose un algorithme de création d'arbre des suffixes en temps linéaire, $O(n)$, où n est la taille de la séquence à indexer [128]. La mémoire nécessaire pour stocker un arbre des suffixes est en $O(n)$ [129]. Dans un arbre des suffixes, on peut tester la présence ou l'absence d'une sous-séquence de taille m dans la séquence indexée. Le temps nécessaire pour cette recherche ne dépend pas de la quantité de données indexées et est en $O(m)$.

Cette structure n'est pas adaptée pour stocker de grandes quantités de séquences. Pour un jeu de données contenant 30 millions de séquences d'ADN de 100 pb, l'équivalent d'un petit jeu de données de Tara Oceans, un arbre des suffixes prendrait 45 Go de mémoire [130]. En conséquence, les arbres des suffixes sont souvent utilisés pour stocker des séquences d'ADN jusqu'à l'échelle du chromosome [129].

2.3.2 Tableau des suffixes

Les tableaux des suffixes ont été introduits en 1990 par Manber et Myers [125]. Cette structure est plus légère en mémoire que l'arbre des suffixes. Comme l'arbre des suffixes, le tableau des suffixes peut être utilisé pour tester la présence d'une sous-séquence dans une séquence, et ce de manière presque aussi efficace que l'arbre des suffixes.

Soit une séquence S composée de n caractères. Un tableau des suffixes de S, appelé T, est un tableau d'entier, de 0 à $n-1$, qui spécifie l'ordre lexicographique des n suffixes de la séquence S. Le suffixe en $T[0]$ est donc, lexicographiquement, le suffixe le plus petit. D'une manière générale, le suffixe en $T[i]$ est lexicographiquement plus petit que le suffixe en $T[i+1]$. Comme pour l'arbre des suffixes, le caractère $ désigne la fin de la séquence. Dans le tableau

Suffixe	$T[]$
$	6
ACTAT$	1
AT$	4
CTAT$	2
T$	5
TACTAT$	0
TAT$	3

TABLE 1: Tableau des suffixes T pour la séquence TACTAT$.

des suffixes, on considère que ce caractère est lexicographiquement plus petit que tous les autres caractères de S : on a donc $T[0] = \$$.

Le tableau 1 représente le tableau des suffixes pour la séquence $S =$ TACTAT$. Rechercher toutes les occurrences d'une sous-séquence s de taille m dans S est équivalent à rechercher s comme étant un préfixe des suffixes de S : si s apparait dans S, il sera forcément au début d'un des préfixes de S. Les suffixes sont ordonnés par ordre lexicographique dans T et les suffixes dont le préfixe est s sont donc contiguës dans T. La recherche de s dans S consiste alors à trouver dans T le premier et le dernier suffixe de S ayant s comme préfixe. Toutes les occurrences entre ces deux suffixes contiennent aussi s.

Les tableaux des suffixes sont plus économes en mémoire que les arbres des suffixes. Pour indexer une séquence de n caractères il faut 40 bits/caractères, soit $5n$ octets de mémoire [125, 129]. Pour indexer 30 millions de séquences d'ADN de 100 pb, soit 3×10^9 nucléotides, l'empreinte mémoire d'un arbre des suffixes est donc de $5 \times 3 \times 10^9$ octets, soit 13,97 Go. Cela est plus faible que les 45 Go de l'arbre des suffixes, pour un temps de construction similaire. Par contre, le temps de recherche d'une sous-séquence dans un tableau des suffixes est en $O(m \log n)$ [129], ce qui est moins bon que l'arbre des suffixes, qui en $O(m)$. D'autres modules peuvent être rajoutés au tableau des suffixes afin de l'accélérer ou de lui rajouter des fonctionnalités plus avancées, par exemple pour mener des recherches en $O(m)$ [131]. Toutes ces modifications alourdissent la structure et ne sont pas présentées ici.

2.3.3 FM-index

Le FM-index est une structure de données compressée, présentée en 2000 par Ferragina et Manzini [126], qui utilise le tableau des suffixes et la transformée de Burrows-Wheeler (ou BWT, pour *Burrows-Wheeler Transform*) [132].

Dans la transformée de Burrows-Wheeler, on calcule toutes les rotations de la séquence à indexer, en décalant la séquence d'un caractère à chaque permutation. Comme pour le tableau des suffixes, ces permutations sont ensuite ordonnées par ordre lexicographique (voir tableau 2b). Le tableau ordonné est donc identique au tableau ordonné des suffixes (voir tableau 1), excepté pour le préfixe issu de la rotation qui est présent après le symbole $.

La dernière colonne du tableau ordonné est la BWT. La première colonne du tableau ordonné contient tous les caractères de la séquences, ordonnés lexicographiquement. Cette information est résumée par la construction du tableau des premières occurrences de chaque caractère de la séquence (voir tableau 2c).

Le FM-index consiste en la transformée de Burrows-Wheeler. À partir de cette seule information, on peut tester la présence ou l'absence d'une sous-séquence dans la séquence indexée

T	A	C	T	A	T	$
A	C	T	A	T	$	T
C	T	A	T	$	T	A
T	A	T	$	T	A	C
A	T	$	T	A	C	T
T	$	T	A	C	T	A
$	T	A	C	T	A	T

(a) Permutations de la séquence TACTAT$.

$	T	A	C	T	A	T
A	C	T	A	T	$	T
A	T	$	T	A	C	T
C	T	A	T	$	T	A
T	$	T	A	C	T	A
T	A	C	T	A	T	$
T	A	T	$	T	A	C

(b) Réorganisation des permutations par ordre lexicographique. La colonne en cyan est la transformée de Burrows-Wheeler.

$	A	C	T
0	1	3	4

(c) Tableau des premières occurrences de chaque nucléotide de la séquence.

TABLE 2: Le FM-index de la séquence TACTAT$ est composé de la transformée de Burrows-Wheeler (colonne cyan du tableau 2b).

en utilisant un algorithme de recherche inversée : la sous-séquence à rechercher est lue en commençant par le dernier caractère.

Pour indexer une séquence S de taille n, la complexité en temps pour la création du FM-index est similaire à celle de l'arbre des suffixes, $O(n)$. Compter le nombre d'occurrences d'une sous-séquence s de taille m dans S se fait en $O(m)$ [129]. L'avantage considérable du FM-index réside dans son empreinte mémoire [133]. L'implémentation d'un FM-index ne nécessite que $2n$ bits [129]. Ainsi, pour stocker 30 millions de séquences de 100 bp, soit 3 milliards de nucléotides, il suffit de $\frac{2\times3\times10^9}{8}$ octets, soit 715,26 Mo. Cette structure de données est donc largement plus avantageuse que l'arbre des suffixes ou le tableau des suffixes.

2.3.4 Table de hachage

Une table de hachage est une structure de données utilisée pour associer une clé à un élément. La recherche d'un élément dans une table de hachage est souvent considérée comme se faisant en temps constant, $O(1)$, bien que dans le pire des cas, cette recherche soit en $O(n)$, où n est le nombre d'éléments indexés dans la table. Pour rechercher si une séquence d'ADN est présente dans un échantillon métagénomique, on peut utiliser cette structure pour indexer toutes les séquences du jeu de données. On peut alors rechercher en temps constant si une séquence a été indexée.

Une table de hachage consiste en un tableau de taille r. Chaque séquence à indexer est transformée en un nombre entre 0 et $r-1$, correspondant à une case donnée du tableau. Dans cette case, on peut stocker un élément. Dans le cas d'un test de présence/absence d'une séquence dans un ensemble de séquence, l'élément à stocker est la séquence elle même (voir figure 10).

La transformation de la séquence en un nombre entre 0 et $r-1$ se fait à l'aide d'une fonction de hachage. Une fonction de hachage est un algorithme qui associe une donnée de taille variable à une donnée de taille fixe, appelée "code de hachage" dans le reste du document. Pour tester si une séquence a été indexée, il suffit de la hacher et de la comparer à la séquence

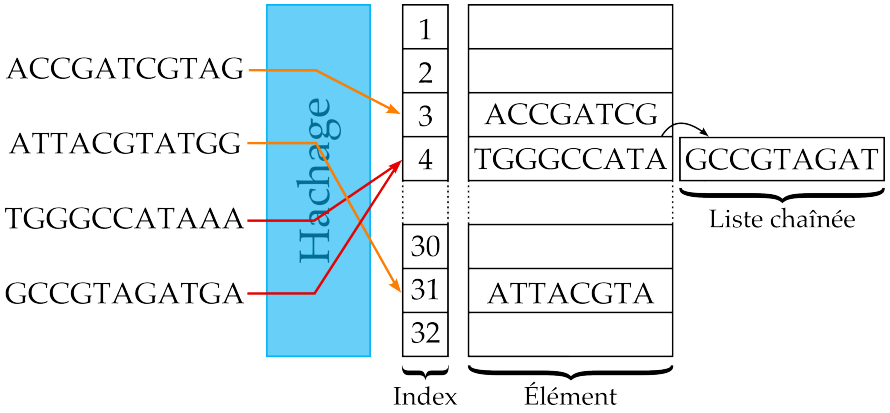


FIGURE 10: Représentation d'une table de hachage stockant la présence de plusieurs séquences. Les deux dernières séquences produisent une collision, résolue par une liste chaînée.

stockée à la position indiquée par le code de hachage. La complexité en temps de la recherche est alors fonction de la taille de la séquence à hacher, et est donc en $O(m)$.

Les tables de hachage sont sujettes aux collisions. Une collision se produit quand la fonction de hachage calcule le même code de hachage pour deux clés différentes (voir les deux dernières séquences de la figure 10). On ne peut pas directement stocker deux éléments différents dans une même case du tableau. Une des solutions pour remédier à cela est d'utiliser une liste chaînée pour stocker les éléments ayant le même code de hachage. Ainsi, plusieurs valeurs différentes mais ayant le même code de hachage peuvent être indexées. Dans ce cas, la recherche n'est plus en temps constant ; la recherche dans une liste chaînée se fait en temps linéaire $O(e)$, où e est le nombre d'éléments dans la liste.

Une table de hachage peut être utilisée pour stocker de manière exacte n séquences de taille m. Une telle structure stocke les séquences explicitement. En utilisant le codage classique de l'ADN en binaire, soit 2 bits par nucléotide, il faut $n \cdot \lceil \frac{2m}{8} \rceil \cdot 8$ bits, soit 715,26 Mo pour stocker 30 millions de séquences de 100 pb ($\lceil \frac{2m}{8} \rceil$ étant l'arrondi à l'octet supérieur du stockage d'une séquence). Pour être tout à fait exact, une séquence ne peut être représentée que sur une puissance de 2 bits. On a alors besoin de $n \cdot 2^{\lceil \log_2(2m) \rceil}$ bits, soit 912,53 Mo pour stocker 30 millions de séquences de 100 pb ($2^{\lceil \log_2(2m) \rceil}$ étant l'arrondi à la puissance de 2 bits supérieur du stockage d'une séquence). Ce calcul ne prend pas en compte les listes chaînées pour pallier les collisions. La taille mémoire est donc légèrement supérieur au FM-index mais le temps de recherche dans la table de hachage est meilleur.

2.3.5 Filtre de Bloom

Le filtre de Bloom est une structure de données probabiliste, et non exacte. Elle peut ainsi retourner une réponse fausse. Il est important de définir les termes de *vrai positif*, *faux positif*, *vrai négatif* et *faux négatif* avant d'analyser cette structure. La figure 11 représente ces notions.

A. VRAI POSITIF : une séquence identifiée (positif) à raison (vrai) par un algorithme dans un ensemble de séquence est appelée un *vrai positif*, il est noté **vp**.

B. FAUX POSITIF : une séquence identifiée (positif) à tord (faux) par un algorithme dans un ensemble de séquence est appelée un *faux positif*, il est noté **fp**.

C. VRAI NÉGATIF : une séquence non-identifiée (négatif) à raison (vrai) par un algorithme dans un ensemble de séquence est appelée un *vrai négatif*, il est noté **vn**.

D. FAUX NÉGATIF : une séquence non-identifiée (négatif) à tord (faux) par un algorithme dans un ensemble de séquence est appelée un *faux négatif*, il est noté **fn**.

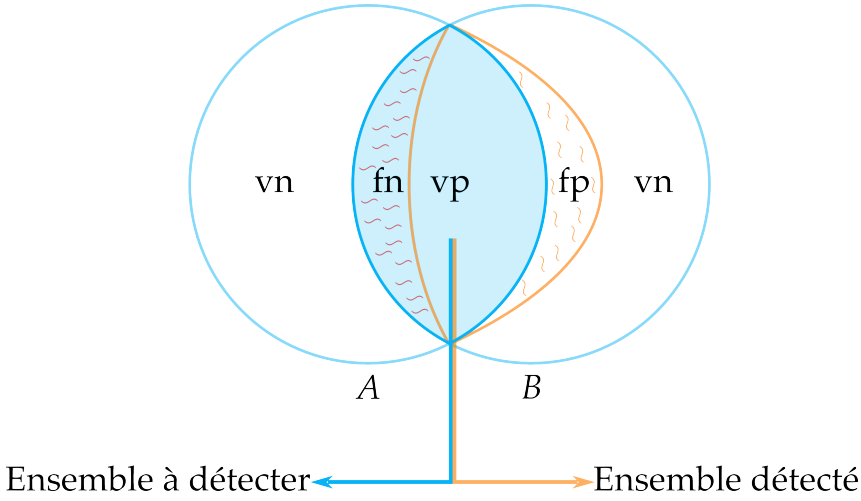


FIGURE 11: Représentation des notions de *vrai positif*, *faux positif*, *vrai négatif* et *faux négatif* sur un algorithme cherchant à identifier les éléments communs à deux ensembles. Les éléments communs aux deux ensemble A et B sont les éléments à détecter et sont représentés sous un fond bleu, à l'intersection de A et B. Les éléments non-communs aux deux ensemble A et B sont représentés sous un fond blanc.
Les éléments communs identifiés à raison par un algorithme sont les vrai positifs (vp) et les éléments communs qui ne sont pas identifiés sont des faux négatifs (fn), représentés par des vagues horizontales rouges. Les éléments non-communs et non-identifiés par l'algorithme sont les vrai négatifs (vn). L'algorithme peut identifier à tord certains des éléments non-communs : ce sont les faux positifs (fp), représentés par des vagues verticales orange. L'ensemble détecté, entouré en orange, comporte les vrai positifs et les faux positifs.

Le filtre de BLOOM est très comparable à une table de hachage. La différence principale est qu'il n'y a aucune gestion des collisions [127]. De plus, le filtre de BLOOM utilise plusieurs fonctions de hachage mais ne stocke pas d'information : seul un booléen est présent dans chaque case. Cette structure a été créée pour tester l'appartenance d'un élément à un ensemble. Les filtres de BLOOM ont été récemment utilisés en bio-informatique, par exemple dans la problématique de l'assemblage [134] où il a été possible de procéder à un assemblage métagénomique *de novo* en utilisant 30 fois moins de mémoire que classiquement.

Le filtre de Bloom consiste en un tableau de q bits, tous initialisés à 0, et h fonctions de hachage. Chaque fonction de hachage permet d'attribuer à un élément une unique position dans le tableau. Ainsi, chaque élément est codé par h positions dans le tableau de bits.

Pour insérer un élément dans le filtre, on le hache avec h fonctions de hachage différentes. Chaque code de hachage correspond à une unique position dans le tableau et le bit à cette position est mis à 1, quelle que soit sa valeur précédente. Pour tester la présence d'un élément dans le filtre de Bloom, on commence par le hacher avec les mêmes h fonctions de hachage que précédemment. On contrôle la valeur des bits ciblés par ces codes de hachage. Si tous les bits sont à 1, on estime que l'élément est présent dans l'ensemble indexé. Si au moins un des bits est à 0, on est certain que l'élément ne se trouve pas dans l'index (voir fig.12).

Cette structure est probabiliste, c'est-à-dire elle génère des faux positifs. La requête de présence d'un élément dans un ensemble retourne soit "l'élément est probablement présent dans l'ensemble", soit "il est sûr que l'élément n'est pas dans l'ensemble". À l'inverse d'une table de hachage, les collisions ne sont pas résolues. Dès lors, le code de hachage d'un élément à rechercher peut être identique au code de hachage d'un élément différent mais effectivement indexé. Un faux positif correspond à un élément non présent dans le filtre mais où ses h bits sont à 1 : le filtre répondra donc, à tord, que l'élément existe.

Par contre, si un élément est bien dans le jeu indexé, tous les bits correspondant dans le filtre seront à 1. Si on recherche cet élément, on ne peut pas trouver un des bits à 0 et il est

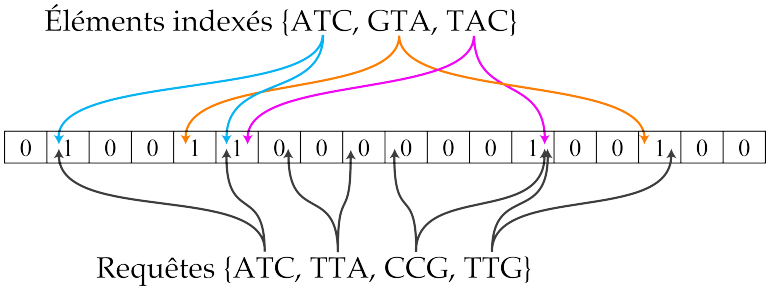


FIGURE 12: Représentation d'un filtre de Bloom utilisant $h = 2$ fonctions de hachage. L'élément TTG est retrouvé à tord : c'est un faux positif.

donc identifié comme présent : tous les vrai positifs sont identifiés comme positifs. De même, si un élément est identifié comme absent de l'ensemble par le filtre de BLOOM, il est certain que cet élément n'en fait pas partie : les faux négatifs ne sont pas possibles.

Le filtre de BLOOM retourne une mauvaise réponse avec une probabilité non nulle. La mesure de cette probabilité est appelée **taux de faux positifs**. Une approximation asymptotique de ce taux de faux positif est $0.6185^{q/n}$, pour n éléments insérés dans le tableau de taille q et $(\ln 2 \cdot (q/n))$ fonctions de hachage utilisées [135].

Les filtres de BLOOM sont un bon moyen pour stocker efficacement de l'information. Broder et Mitzenmacher [135] ont montré qu'il suffit de $(n \log_2 e \cdot \log_2(1/\varepsilon))$ bits pour permettre une requête de présence / absence parmi n éléments, avec un taux de faux positifs égal à ε. La taille des éléments à indexer n'a pas d'impact sur la taille mémoire utilisée. Ainsi, pour indexer 30 millions de séquences, avec un taux de faux positif de 0,1%, il suffit de 51,42 MO.

Il est important de noter que cette structure ne permet pas de retrouver les éléments indexés à partir du filtre. Il est seulement possible de savoir, avec une probabilité p, si un élément donné y a été indexé. De plus, comme les fonctions de hachage peuvent générer des collisions, un bit peut servir pour plusieurs éléments différents : il n'est donc pas possible d'enlever un élément du filtre sans risquer d'enlever d'autres éléments. Cette structure est plus légère en mémoire que les autres structures présentées ici. La complexité en temps de l'indexation d'un élément et de la recherche d'un élément sont $O(h \times m)$, où h est le nombre de fonctions de hachages utilisées dans le filtre de BLOOM et m la taille des éléments. Cette complexité est indépendante du nombre d'éléments déjà indexés dans la structure.

Conclusion sur les structures de données existantes

Plusieurs structures de données peuvent être utilisées pour indexer un grand nombre de séquences d'ADN, puis rechercher si une séquence donnée est présente dans cet index. Parmi ces structures, le filtre de BLOOM est la structure la plus légère en mémoire (voir tableau 3). Cette structure a par contre un désavantage majeur comparée aux autres présentées ici : elle peut générer des faux positifs, c'est à dire identifier à tord un élément comme étant présent dans l'index. Le nombre de faux positif est intrinsèquement lié à la quantité de mémoire utilisée par le filtre et le nombre d'éléments qui y sont insérés.

Finalement, il faut noter qu'avec l'arbre des suffixes, le tableau des suffixes et le FM-index on peut rechercher n'importe quelle sous-séquence dans la séquence indexée. La table de hachage et le filtre de BLOOM indexent des séquences et permettent seulement la recherche d'une séquence dans l'index, et non d'une sous-séquence.

	Complexité en temps de la recherche	Mémoire utilisée
Arbre des suffixes	$O(m)$	45 Go
Tableau des suffixes	$O(m \log n)$	13,96 Go
FM-index	$O(m)$	715,25 Mo
Table de hachage	$O(m)$	912,53 Mo
Filtre de Bloom	$O(h \times m)$	51,42 Mo

TABLE 3: Caractéristiques de temps de recherche et de mémoire des différentes structures de données présentées. La complexité est celle en temps de la recherche d'une séquence ou d'une sous-séquence dans la structure. La mémoire est celle utilisée par les structures pour 30 millions de séquences de 100 pb. m est la taille de la séquence à rechercher dans l'index, n est le nombre de caractères indexés et h est le nombre de fonctions de hachage utilisées.

2.4 CONCLUSION

Les récents développements en matière de séquençage à haut débit mènent à l'obtention de millions de courtes séquences d'ADN à partir d'un échantillon biologique. De plus, contrairement à la génomique, ce matériel biologique n'a plus besoin d'être cultivé en laboratoire : il peut être directement prélevé dans un milieu naturel. Une nouvelle discipline, la métagénomique, est donc actuellement en plein essor et consiste à étudier ces séquences d'ADN. Plusieurs angles d'attaques sont possibles, mais la plupart nécessite d'avoir un catalogue d'espèces ou de gènes de référence, donc connus et étudiés. Les séquences obtenues sont comparées à ces références. On en déduit alors quelles espèces ou fonctions biologiques sont présentes dans l'échantillon et à quelle proportion.

Une autre famille métagénomique, dite comparative *de novo*, consiste à n'utiliser aucune donnée autre que les jeux de données à comparer. Le but est alors de comparer les séquences de plusieurs échantillons pour trouver quelles séquences sont, par exemple, les plus proches les unes des autres. À partir de ces séquences communes, on peut calculer une distance entre plusieurs jeux. Cette distance reflète alors une distance en terme de composition en espèces : plus deux échantillons ont des séquences communes, plus les espèces qui les composent sont similaires et plus ils sont proches d'un point de vue biologique. Cette comparaison deux à deux peut alors aider à comprendre comment un changement d'environnement influe sur la composition des métagénomes. De plus, il est possible de cibler seulement les séquences communes entre plusieurs échantillons pour initier des analyses plus coûteuses sur seulement une fraction des séquences, par exemple un assemblage métagénomique.

De plus en plus de jeux de données métagénomiques sont composés d'une centaine de millions de séquences d'ADN, généralement de quelques centaines de paires de bases chacune. Pour comparer deux jeux de données de cette taille, il est nécessaire de passer par une phase d'indexation, pour diminuer les temps de recherche dans ces données. Plusieurs structures existent pour cette tâche, mais différents critères, essentiellement temps, vitesse et exactitude des résultats, les différencient. L'approche développée durant cette thèse se base sur le filtre de Bloom. Ce filtre permet d'indexer des données et de rechercher un élément dans cet index en utilisant une quantité de mémoire qui n'est liée ni à la quantité d'éléments à indexer, ni à la taille de ces éléments. En revanche, cette structure génère des faux positifs, c'est à dire qu'elle peut identifier des éléments comme étant présents dans l'index quand bien même ils n'y sont pas.

La structure de données développée durant cette thèse sert de structure d'indexation pour la méthodologie de comparaison de jeux de données métagénomiques mise en place durant ce doctorat. Cette méthodologie, détaillée dans le chapitre 3, calcule deux score de similarité

entre deux échantillons métagénomiques. Il est montré, chapitre 4, que ce score est biologiquement pertinent et permet de discriminer des échantillons provenant de conditions environnementales différentes.

La métagénomique comparative *de novo* est une discipline récente dans laquelle peu de solutions bio-informatiques existent à l'heure actuelle. Notre méthodologie est capable de chercher de l'information biologique pertinente en se basant exclusivement sur les données séquencées. De plus, en utilisant une nouvelle structure, sa consommation mémoire et son temps de calcul autorisent son utilisation sur n'importe quel ordinateur disposant de quelques de Go de mémoire.

3

COMPARAISONS INTENSIVES DE DONNÉES MÉTAGÉNOMIQUES

Dans ce chapitre, nous proposons une nouvelle méthode dédiée à la comparaison de grands jeux de données métagénomiques. La première section (voir 3.1) explique nos motivations et introduit plusieurs définitions utiles pour la suite du manuscrit.

La seconde section (voir 3.2) propose une manière de comparer deux échantillons en introduisant un nouveau score de similarité (voir sous-section 3.2.1). Ce score est obtenu en réalisant l'intersection de deux jeux de données métagénomique. Cette intersection en détaillée sous-section 3.2.2 et génère des faux positifs. Ceux-ci sont analysés et discutés sous-section 3.2.3.

La dernière section (voir 3.3) présente la mise en œuvre de notre méthodologie. Dans cette section, nous détaillons le fonctionnement d'une nouvelle structure de données développée durant ce doctorat, le BDS (voir sous-section 3.3.1). Cette structure indexe efficacement de très grandes quantités de données, *i.e.* plusieurs millions de séquences d'ADN de quelques centaines de paires de bases et autorise une requête de type présence/absence d'un k-mer dans un ensemble de séquence. Finalement, la sous-section 3.3.2 présente COMPAREADS, une implémentation de notre méthodologie qui utilise le BDS comme structure d'indexation.

3.1 PRÉLUDE ET DÉFINITIONS

L'enjeu principal de la métagénomique comparative *de novo* est de comparer des jeux de données sur la seule base de leur contenu brut. Obtenir un score de similarité entre deux échantillons est une façon de les comparer. Lorsque l'on étudie plus de deux échantillons, ce score de similarité permet de faire de la classification entre les différents métagénomes, en regroupant ensemble les échantillons les plus proches. Cette mesure de similarité peut être représentée par le nombre, ou le pourcentage, de séquences identiques ou similaires entre deux jeux de données. Cette mesure est purement *de novo*, elle ne requiert aucune connaissance préalable des génomes présents dans l'échantillon : elle est seulement basée sur les données brutes.

Certains outils de génomique sont utilisés pour identifier des séquences proches : les plus connus sont les logiciels d'alignement tels BLAST, UBLAST ou BLAT. L'utilisation du logiciel BLAST pourrait fournir une information équivalente à la comparaison brute de deux métagénomes. Il faudrait aligner toutes les séquences d'un échantillon sur les séquences d'un second échantillon et faire cela pour tous les échantillons contre tous. BLAST n'a pas été conçu pour cette utilisation. Il ne permet pas d'exécuter cette tâche en un temps et une utilisation mémoire raisonnables sur les jeux de données métagénomiques produits actuellement par les séquenceurs de nouvelle génération. De même, les logiciels BLAT et UBLAST ne passent pas à l'échelle quand le nombre de séquences à comparer devient très important.

3.1.1 Motivations

La quantité de données à traiter en métagénomique est bien plus importante qu'en génomique classique. UBLAST est environ 300 fois plus rapide que BLAST, et BLAT environ 500 fois. Malgré ces vitesses, l'auteur de BLAT estime qu'il faudrait 12 jours, en utilisant 100 processeurs, pour aligner les séquences brutes de deux génomes de souris [103]. Comparer ainsi deux métagénomes, pouvant contenir des milliers d'espèces différentes, n'est donc pour le moment pas réalisable en un temps raisonnable avec ces approches. De plus, les gros projets actuels génèrent un grand nombre d'échantillons. Par exemple, le projet TARA OCEANS va générer environ 2000 échantillons métagénomiques contenant chacun plus de 100 millions de séquences, qu'il faudrait comparer les unes contre les autres.

Les jeux de données métagénomiques actuels nécessitent d'utiliser des structures de données optimisées et adaptées aux questions posées. Dans le cadre de la comparaison de deux métagénomes sur la base du nombre de séquences d'ADN qu'ils partagent, on cherche à tester la présence de chaque séquence d'un jeu de données dans un autre jeu. Pour deux jeux de données de 100 millions de séquences chacun, c'est donc 200 millions de séquences que l'on doit tester ; l'opération élémentaire, qui teste si une séquence est présente dans un ensemble de séquence, est au cœur de la méthode et doit être la plus rapide possible.

Un deuxième point important de la comparaison de deux jeux de données est l'empreinte mémoire. Pour comparer deux jeux, l'un des jeux doit être indexé, de manière à tester rapidement la présence ou l'absence de chaque séquence du second jeu. Or, comme expliqué dans l'état de l'art (voir section 2.3) plusieurs structures de données sont disponibles. Parmi elles, le filtre de BLOOM est la structure la plus légère en mémoire.

Le filtre de BLOOM résout nos deux problèmes majeurs : avoir une structure d'indexation légère en mémoire et surtout offrant la possibilité de réaliser des tests de présence très rapides et en un temps indépendant de la quantité de données indexées.

3.1.2 Définitions

Avant de rentrer dans le détail de la méthode développée durant ce doctorat, il est important de formaliser quelques définitions.

A. SÉQUENCE : Une *séquence* est composée de zéro ou plus de symboles appartenant à un alphabet Σ. Ici, le travail porte sur de l'ADN. L'alphabet est alors $\Sigma = \{A,C,G,T\}$. Une séquence s de longueur m sur Σ peut alors se définir comme $s[0]s[1]s[2]...s[m-1]$ avec $s[i] \in \Sigma$ pour $0 \leq i < m$. La longueur d'une séquence s est représentée par $|s|$.

B. K-MER : On note $s[i,j]$ la *sous-séquence* $s[i]s[i+1]...s[j]$ de s. On dit alors que cette sous-séquence de s apparait à la position i. On appel *k-mer* une séquence de longueur k et $s[i,i+k-1]$ est un k-mer de s apparaissant à la position i.

C. K-MER PARTAGÉ : Deux séquences s_1 et s_2 *partagent* un k-mer si et seulement si $\exists(i_1,i_2)$ tel que $s_1[i_1,i_1+k-1] = s_2[i_2,i_2+k-1]$

D. SÉQUENCES SIMILAIRES : Dans la suite de ce manuscrit, on dira que deux séquences s_1 et s_2 sont *similaires* si et seulement si, pour deux entiers k et t, ces deux séquences partagent au moins t k-mers non chevauchants.

3.2 MÉTHODOLOGIE

Cette section s'attache à décrire la méthodologie utilisée pour comparer deux jeux de données métagénomiques issus des NGS. Dans le premier point, présenté sous-section 3.2.1, on

définit deux notions de score de similarité. Ces scores ne sont pas basés sur un résultat d'alignement, mais sur un concept de similarité approximative d'un jeu de données par rapport à un second jeu de données.

Les scores de similarité sont obtenus par une opération centrale de notre méthodologie, définie comme une intersection de deux jeux de données. Les principes et conséquences de l'intersection de jeux de données sont présentés sous-section 3.2.2.

L'intersection de deux jeux de données conduit à la création de faux positifs. La sous-section 3.2.3 discute de ces faux positifs et des moyens mis en place pour les éviter au maximum.

3.2.1 Score de similarité

Pour comparer deux échantillons, il est nécessaire de disposer d'une mesure. Plusieurs mesures différentes peuvent être utilisées, par exemple le taux de GC ou un score d'alignement. Dans notre méthode, nous utilisons une autre mesure, basée sur le nombre de séquences dites "similaires" entre deux jeux de données.

3.2.1.1 *Une notion de score sans alignement*

Plusieurs logiciels réalisent un alignement entre deux séquences et délivrent un score. En alignant toutes les séquences d'un jeu de données sur les séquences d'un second jeu, on peut alors déduire un score de similarité entre les deux jeux. Ce score reflète le nombre de séquences "correctement" alignées, suivant plusieurs critères. Le problème majeur de cette approche est le coût que représente l'alignement de deux séquences. Ce processus est complexe et doit être répété pour chaque séquence. Dans le cadre de très gros échantillons métagénomiques, ce processus d'alignement doit être réalisé plusieurs centaines de millions de fois. Dès lors, les approches basées sur l'alignement sont trop longues et trop coûteuses en ressources.

Quand on aligne deux séquences, on cherche généralement à savoir si ces deux séquences sont identiques ou similaire, c'est-à-dire presque identiques. Si calculer un alignement entre deux séquences est coûteux, rechercher si deux séquences sont identiques ou similaires peut être fait de manière plus simple et plus rapide. Deux séquences similaires ont une ou plusieurs régions identiques de nucléotides. On peut alors rechercher si deux séquences ont de telles régions en communs, c'est-à-dire si deux séquences comportent des sous-séquences strictement identiques.

La méthode développée pendant ce doctorat identifie toutes les séquences similaires entre deux jeux de données. Si cette opération peut sembler basique, elle doit être extrêmement efficace en terme de temps de calcul et de mémoire afin de pouvoir supporter les grandes quantités de données générées par les approches NGS en métagénomiques. Afin d'être performant, on utilise une notion de similarité grossière mais efficace telle que définie sous-section 3.1.2-D : deux séquences seront dites similaires si elles partagent au moins t k-mers distincts.

3.2.1.2 *Scores de similarité entre échantillons*

Notre méthode permet de calculer deux scores de similarité, comme expliqué dans les deux parties suivantes. Ces deux scores sont basés sur le nombre de séquences d'un échantillon A similaires à au moins une séquence d'un échantillon B, noté $(A \overset{\rightsquigarrow}{\cap} B)$, et sur le nombre de séquences de B similaires à au moins une séquence de A, noté $(B \overset{\rightsquigarrow}{\cap} A)$.

SCORE DE SIMILARITÉ D'UN JEU DE DONNÉES Notre approche génère un premier score de similarité, pour chacun des deux jeux de données comparés. Le score est calculé comme étant le nombre de séquences d'un premier jeu similaires à des séquences d'un second jeu, di-

visé par le nombre total de séquences du premier jeu. L'équation 5 donne le score de similarité de l'échantillon A contre l'échantillon B.

$$S(A_B) = \frac{\left|(A \overset{\rightsquigarrow}{\cap} B)\right|}{|A|} \times 100 \tag{5}$$

où $|X|$ est la cardinalité de l'ensemble X,
dans notre cas le nombre de séquences

Comme expliqué sous-section 2.2.3, avoir un score de similarité par échantillon est plus précis qu'un seul score de similarité par intersection, comme utilisé dans GOS. Dans l'étude GOS, la similarité de A contre B est identique à la similarité de B contre A ; dans notre méthode, le score de similarité $S(A_B)$ issu de $(A \overset{\rightsquigarrow}{\cap} B)$ peut être différent du score de similarité $S(B_A)$ issu de $(B \overset{\rightsquigarrow}{\cap} A)$.

SCORE DE SIMILARITÉ GLOBAL Notre approche génère un second score de similarité qui est global et symétrique entre les deux échantillons comparés. Ce score est calculé en divisant le nombre total de séquences similaires trouvées, par le nombre total de séquences, tel que défini équation 6.

$$S(A,B) = \frac{\left|(A \overset{\rightsquigarrow}{\cap} B)\right| + \left|(B \overset{\rightsquigarrow}{\cap} A)\right|}{|A| + |B|} \times 100 \tag{6}$$

Ce score s'apparente à la distance de JACCARD [136], utilisée traditionnellement pour mesurer la diversité et la similarité de deux ensembles (voir équation 7). La différence est que le score obtenu ici se fait sur des multiensembles au lieu de simples ensembles ; contrairement à un ensemble, un multiensemble peut contenir plusieurs fois le même élément.

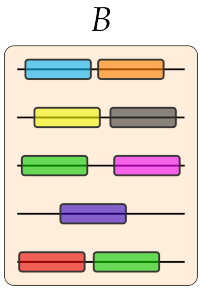
$$JS(A,B) = \frac{|A \cap B|}{|A \cup B|} \times 100 \tag{7}$$

3.2.2 Comparaison par intersection

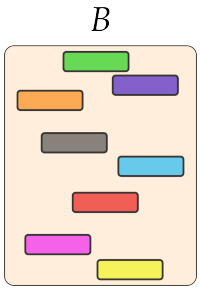
Comme expliqué dans la sous-section précédente, notre méthode conduit à l'obtention de plusieurs scores de similarité. Ces scores sont basés sur le nombre de séquences similaires. Notre méthode consiste donc à réaliser des intersections entre deux jeux de données. Une intersection sert à identifier les séquences d'un premier jeu de données, similaires à des séquences d'un second jeu de données.

3.2.2.1 *Similarité approximative*

Soit deux jeux de données A et B composés d'un ensemble de séquences. La fonction principale de notre méthode est de trouver le sous-ensemble de séquences de A similaires à au moins une séquence de B. Le résultat exact de cette recherche est noté $(A \overrightarrow{\cap} B)$. On utilise un filtre de BLOOM, une structure de données probabiliste, pour effectuer cette opération ; le résultat de la comparaison est alors une sur-représentation de $(A \overrightarrow{\cap} B)$ pouvant contenir des faux positifs (voir sous-sections 3.2.3 et 3.3.1.2) et sera donc noté $(A \overset{\rightsquigarrow}{\cap} B)$. On remarque que $(A \overrightarrow{\cap} B) \subseteq (A \overset{\rightsquigarrow}{\cap} B)$. Le calcul de $(A \overset{\rightsquigarrow}{\cap} B)$ se passe en deux étapes : l'indexation et la requête. Ces deux étapes sont expliquées en détail sous-section 3.2.2.2.



(a) Échantillon non-indexé. Les deux *k*-mers verts sont identiques mais les séquences les portant sont différentes.



(b) Échantillon indexé dans le filtre de Bloom. Chaque *k*-mer est unique et n'est plus rattaché à une séquence.

FIGURE 13: Représentation de l'échantillon B avant (13a) et après (13b) son indexation dans le filtre de Bloom. Des *k*-mers de chaque séquence sont représentés par des rectangles de couleur. Sur la figure 13a, les *k*-mers sont assignés aux séquences auxquelles ils appartiennent.

Le résultat de notre méthode est composé des deux jeux de données $(A \overrightarrow{\cap} B)$ et $(B \overrightarrow{\cap} A)$, contenant respectivement les séquences de A similaires à des séquences de B et les séquences de B similaires à des séquences de A.

3.2.2.2 *Fonction de base : une demi-intersection $A \overrightarrow{\cap} B$*

Soit deux jeux de données A et B, composés de plusieurs séquences. La fonction de base de notre méthode, identifier toutes les séquences de A étant similaires à au moins une séquence de B, est réalisée en deux étapes distinctes : l'indexation dans le filtre de Bloom et la phase de requête.

Le filtre de Bloom utilise une quantité de mémoire fixe. Dans une quantité de mémoire donnée, on peut insérer n'importe quelle quantité de *k*-mers. Plus la quantité de *k*-mers indexés augmente, plus la probabilité p d'obtenir un faux positif sur le test de présence d'un *k*-mer dans l'index est élevé. Il n'y a par contre pas de faux négatifs.

INDEXATION Le jeu de données B est indexé dans le filtre de Bloom : tous les *k*-mers de toutes les séquences de B sont indexés dans la structure de données (voir algorithme 1). À la fin de cette phase d'indexation, le filtre de Bloom indexe donc tous les *k*-mers ayant au moins une occurrence dans le jeu B. Dans la suite de ce document, les échantillons sont représentés de deux façons différentes suivant qu'ils sont utilisés pour la phase de requête ou la phase d'indexation (voir figure 13) : les *k*-mers indexés ne contiennent pas de doublons et ne sont pas assignés à une séquence.

```
Données : Un ensemble de séquences B.

début
    pour chaque Séquence seq de B faire
        pour chaque k-mer de seq faire
            Indexer(k-mer)
        fin
    fin
fin
```

Algorithme 1: Phase Indexation(B) d'un ensemble de séquences.

REQUÊTE La phase de requête traite toutes les séquences de A une par une. Pour une séquence $s \in A$, l'index est utilisé pour tester dans B la présence de chaque k-mer de s (voir algorithme 2). Si s partage au moins t k-mers **non-chevauchants** avec l'index, on considère que s est similaire à une séquence de B ; la séquence s est alors insérée dans $(A \overrightarrow{\cap} B)$ (voir figure 14). Si s ne partage pas au moins t k-mers non-chevauchants avec l'index, elle n'est pas similaire à une séquence de B et n'est pas insérée dans le résultat (voir figure 15). Le calcul d'une demi-intersection est présenté par l'algorithme 3. La figure 16 représente la phase de requête pour $t = 2$ entre le jeu A, et le jeu B indexé dans le filtre de BLOOM.

```
Données : Un ensemble de séquences A, un index et le paramètre 't'.
Résultat : Un ensembles de séquences (A ⃗∩ B).

début
    pour chaque Séquence seq de A faire
        i = 0
        tant que i < (|seq|-k+1) faire
            si EstDansIndex(seq[i, i+k-1]) alors
                nb += 1
                i += k
            fin
            si nb >= t alors
                (A ⃗∩ B) += seq
                stop
            fin
            sinon
                i += 1
            fin
        fin
    fin
    retourner (A ⃗∩ B)
fin
```

Algorithme 2: Phase Requête(A) d'un ensemble de séquences.

```
Données : Deux ensembles de séquences A et B.
Résultat : Un ensembles de séquences (A ⃗∩ B).

début
    Indexation(B)
    (A ⃗∩ B) = Requête(A)
    retourner (A ⃗∩ B)
fin
```

Algorithme 3: Calcul de demi-intersection(A, B) qui retourne (A $\overrightarrow{\cap}$ B).

PLUSIEURS ÉTAPES D'INDEXATION Comme expliqué précédemment, plus la quantité de k-mers indexés dans le filtre de BLOOM est grande, plus la probabilité p d'obtenir un faux positif pour un k-mer lors de la phase de requête est grande. Afin de contrôler le taux de faux positif, au maximum l k-mers peuvent être indexés simultanément dans le filtre de BLOOM.

Le calcul complet d'une demi-intersection se déroule alors en plusieurs phases d'indexations et de requêtes (voir figure 17). La première phase d'indexation rempli le filtre de BLOOM

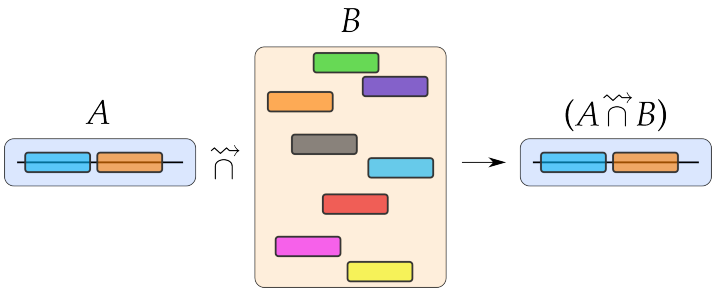


FIGURE 14: Exemple de séquence de A retrouvée dans l'index B pour $t = 2$. Dans le jeu A, la séquence est représentée par une ligne horizontale. Dessus, deux k-mers sont représentés par des rectangles. L'index B contient plusieurs k-mers. La séquence de A partage au moins deux k-mers avec l'index de B, elle est placée dans l'ensemble $(A \overset{\rightsquigarrow}{\cap} B)$.

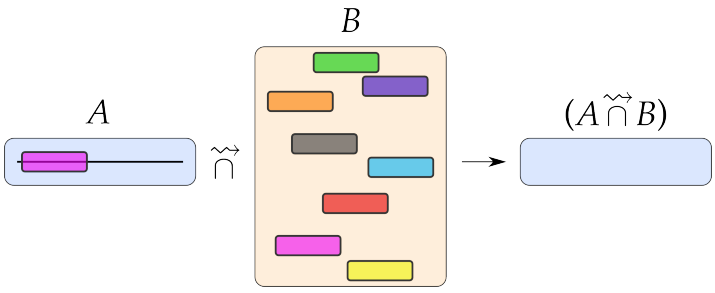


FIGURE 15: Exemple de séquence de A non retrouvée dans l'index B pour $t = 2$. Dans le jeu A, la séquence est représentée par une ligne horizontale. Dessus, un k-mer est représenté par un rectangle mauve. L'index B contient plusieurs k-mers. La séquence de A n'ayant pas au moins deux k-mers partagés avec l'index de B, elle ne fait pas partie de l'ensemble $(A \overset{\rightsquigarrow}{\cap} B)$.

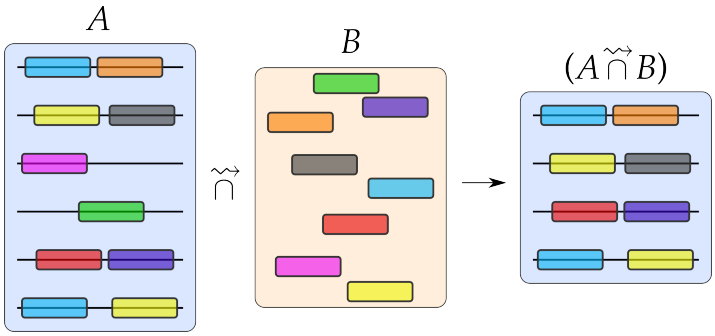


FIGURE 16: Représentation de la phase de requête entre les échantillons A et B pour $t = 2$. Dans le jeu A, les séquences sont représentées par une ligne horizontale. Sur chaque séquence, un ou deux k-mers sont représentés par des rectangles. Seuls les k-mers partagés entre les deux jeux sont représentés. Les séquences de A partageant au moins deux k-mers avec l'index de B sont placées dans l'ensemble $(A \overset{\rightsquigarrow}{\cap} B)$.

avec les l premiers k-mers de B. Suit alors la phase de requête, qui est menée en utilisant toutes les séquences de A. À la fin de ce premier calcul, $(A \overrightarrow{\cap} B)$ ne contient qu'une demi-intersection partielle, noté $(A \overrightarrow{\cap} B)^{tmp}$, et composée des séquences de A similaires à des séquences indexées de B. Le filtre de Bloom est alors réinitialisé et une seconde phase d'indexation est menée avec les l k-mers suivants de B. Suit la phase de requête utilisant toutes les séquences de A, qui conduit à une seconde demi-intersection partielle. Le processus est répété séquentiellement jusqu'à ce que toutes les k-mers de B aient été indexés une fois.

En terme de vrais positifs, ce partitionnement est strictement équivalent à indexer complètement le jeu B puis à traiter toutes les séquences de A sur cet index. Afin d'éviter les calculs redondants, les séquences de A identifiées comme similaires dans l'une des intersections partielles sont marquées comme telles dans un vecteur de bits et ne seront plus utilisées lors des phases de requête suivantes.

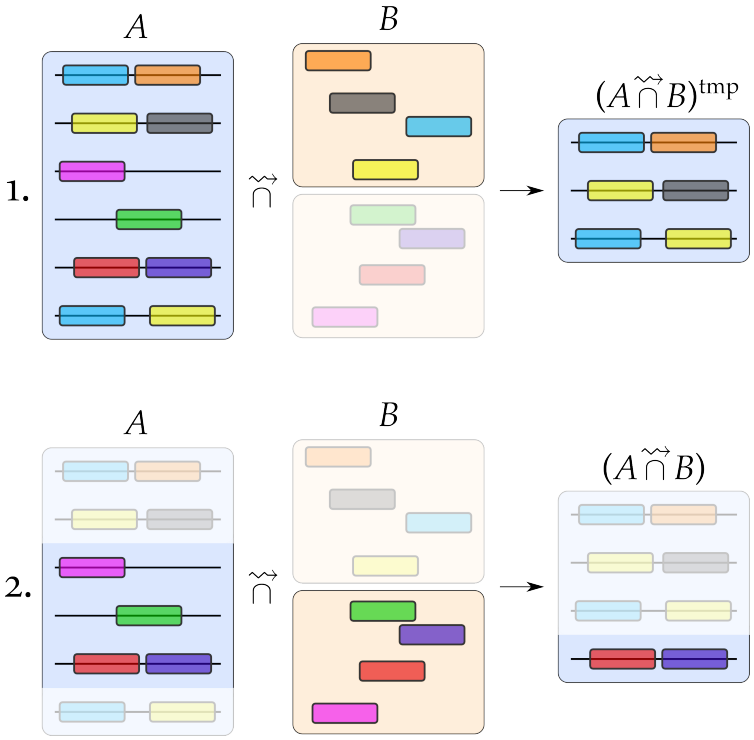


FIGURE 17: Représentation de la phase de requête entre les échantillons A et B pour $t = 2$. Contrairement à la figure 16, l'échantillon B est ici indexé en deux temps. À la fin de la première étape, $(A \overrightarrow{\cap} B)$ n'est pas complet et est alors noté $(A \overrightarrow{\cap} B)^{tmp}$. Les séquences de A déjà identifiées comme similaires ne sont pas réutilisées lors de la seconde étape.

COMPLEXITÉ EN TEMPS Soit n_A et n_B le nombre de k-mers respectivement dans le jeu de données A et B. Calculer $A \overrightarrow{\cap} B$ se fait en $O(n_B)$ (indexation) $+ O(n_A \times \lceil \frac{n_B}{l} \rceil)$ (requête). Le terme $\lceil \frac{n_B}{l} \rceil$ vient de la limitation du remplissage du filtre de Bloom pour maintenir un taux de faux positif constant. Dans le périmètre d'application des travaux de cette thèse, $\lceil \frac{n_B}{l} \rceil$ est généralement de l'ordre de quelque dizaines maximum.

3.2.2.3 *Pipeline complet de la méthode*

Calculer $A \overrightarrow{\cap} B$ est asymétrique : cela ne permet pas d'obtenir l'intersection des échantillons A et B, mais seulement les séquences de A similaires à des séquences de B. Pour obtenir les séquences de B similaires à des séquences de A, il est nécessaire de calculer $B \overrightarrow{\cap} A$. En pratique, pour comparer complètement et symétriquement deux échantillons A et B, un pipeline en trois

étapes, plus compliqué que simplement $A \overrightarrow{\cap} B$ suivi de $B \overrightarrow{\cap} A$, est effectué (voir algorithme 4). Ces trois étapes permettent de réduire les faux positifs dans les deux demi-intersections.

TROIS ÉTAPES Le pipeline complet de la méthode pour calculer $A \overrightarrow{\cap} B$ et $B \overrightarrow{\cap} A$ est le suivant :

1. Calculer $A \overrightarrow{\cap} B$, les séquences trouvées sont stockées dans un sous ensemble noté $(A \overrightarrow{\cap} B)^*$ (voir figure 18a),
2. Calculer $B \overrightarrow{\cap} (A \overrightarrow{\cap} B)^*$, les séquences trouvées sont stockées dans le sous ensemble $(B \overrightarrow{\cap} A)$ (voir figure 18b),
3. Calculer $A \overrightarrow{\cap} (B \overrightarrow{\cap} A)$, les séquences trouvées sont stockées dans le sous ensemble $(A \overrightarrow{\cap} B)$ (voir figure 18c).

Les deux jeux de données en sortie du processus, $(A \overrightarrow{\cap} B)$ et $(B \overrightarrow{\cap} A)$, sont ainsi obtenus en appliquant l'opération fondamentale $\overrightarrow{\cap}$ entre un jeu requête et un jeu indexé résultant déjà d'une opération $\overrightarrow{\cap}$ asymétrique. Il est important de noter qu'en pratique, la dernière intersection permettant d'obtenir $(A \overrightarrow{\cap} B)$ est calculée par $(A \overrightarrow{\cap} B)^* \overrightarrow{\cap} (B \overrightarrow{\cap} A)$ au lieu de simplement $A \overrightarrow{\cap} (B \overrightarrow{\cap} A)$ comme indiqué précédemment afin de simplifier la lecture. Ces deux calculs produisent exactement le même résultat en terme de vrais positifs, mais $(A \overrightarrow{\cap} B)^* \overrightarrow{\cap} (B \overrightarrow{\cap} A)$ est plus rapide que $A \overrightarrow{\cap} (B \overrightarrow{\cap} A)$ car $\left|(A \overrightarrow{\cap} B)^*\right| \leq |A|$. Les séquences non-similaires de la première étape ne sont pas re-testées inutilement (voir figure 18). Comme expliqué dans la sous-section suivante, ces trois étapes permettent de retirer une partie des faux positifs.

Données : Deux ensembles de séquences A et B.
Résultat : Deux ensembles de séquences (A $\overrightarrow{\cap}$ B) et (B $\overrightarrow{\cap}$ A).

début
  $(A \overrightarrow{\cap} B)^*$ = `Demi-intersection(`A, B`)`
  $(B \overrightarrow{\cap} A)$ = `Demi-intersection(`B, $(A \overrightarrow{\cap} B)^*$`)`
  $(A \overrightarrow{\cap} B)$ = `Demi-intersection(`$(A \overrightarrow{\cap} B)^*$, $(B \overrightarrow{\cap} A)$`)`
fin

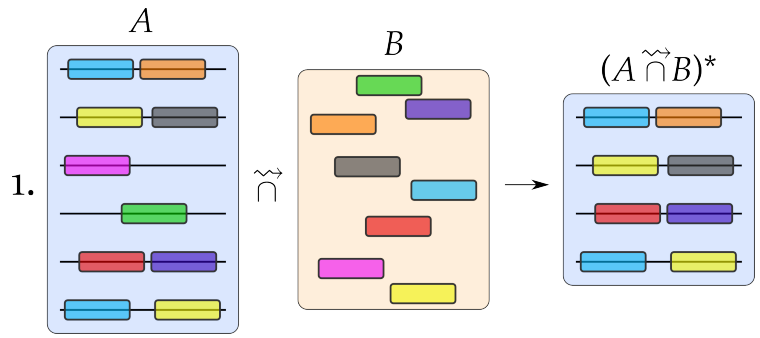
Algorithme 4: Fonctionnement de la méthode pour calculer les deux demi-intersections (A $\overrightarrow{\cap}$ B) et (B $\overrightarrow{\cap}$ A).

3.2.3 Différents type de faux positifs

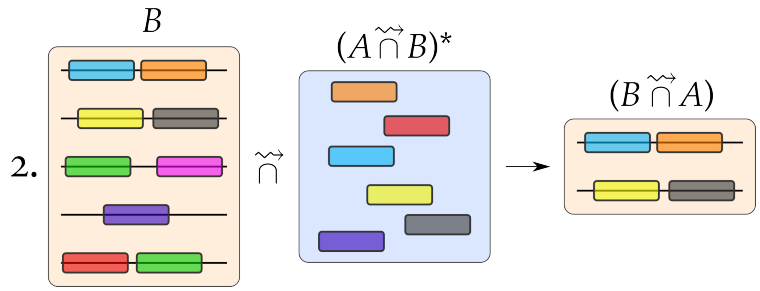
L'approche employée dans notre méthode peut générer des faux positifs pour deux raisons distinctes. Dans les parties suivantes, ces deux sources de faux positifs vont être discutées.

3.2.3.1 *Faux positifs du filtre de* Bloom

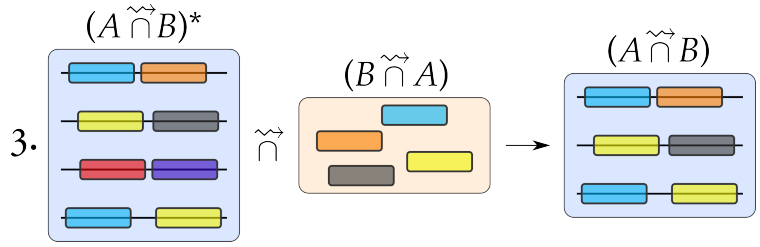
Comme expliqué dans l'état de l'art (voir 2.3.5), le filtre de Bloom génère des faux positifs. En effet, la requête de présence d'un élément dans un filtre de Bloom retourne soit "l'élément est probablement présent dans l'ensemble", soit "il est sûr que l'élément n'est pas dans l'ensemble". À l'inverse d'une table de hachage, les collisions ne sont pas résolues. Dès lors, le code de hachage d'un élément à rechercher peut être identique au code de hachage d'un élément différent mais effectivement indexé. Ces faux positifs sont étudiés en détail dans la sous-section 3.3.1.2.



(a) Première étape, calcul des séquences similaires entre A et B. Ce calcul donne une intersection temporaire $(A \overrightarrow{\cap} B)^*$.



(b) Seconde étape, calcul des séquences similaires entre B et le résultat temporaire de la première demi-intersection. Ce calcul donne la demi-intersection $(B \overrightarrow{\cap} A)$.



(c) Troisième étape, calcul des séquences similaires entre l'intersection temporaire de la première étape et le résultat de la seconde étape. Ce calcul donne la demi-intersection $(A \overrightarrow{\cap} B)$.

FIGURE 18: Le pipeline complet de la méthode permet le calcul de $(A \overrightarrow{\cap} B)$ et $(B \overrightarrow{\cap} A)$ en suivant les trois étapes décrites dans cette figure. On voit que la dernière étape permet d'enlever un faux positif de séquence présent dans $(A \overrightarrow{\cap} B)^*$. La résolution de ce cas de faux positif est expliqué en détail figure 20.

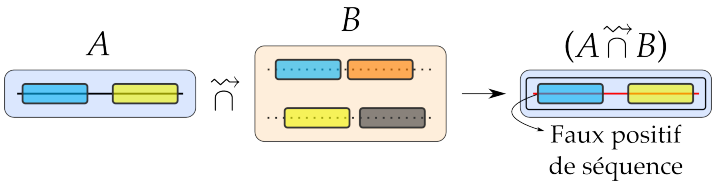


FIGURE 19: Représentation d'un faux positif de séquence. Les k-mers des séquences indexées dans le filtre de BLOOM ne sont normalement pas rattachés à une séquence (voir figure 13). Pour faciliter la vision des faux positifs de séquence, les k-mers partagés qui sont indexés sont ici représentés sur leur séquence d'appartenance en pointillés gris.

3.2.3.2 *Faux positif de séquence*

En utilisant $t > 1$, notre méthode peut identifier une séquence comme similaire bien qu'elle ne respecte pas strictement la définition de similarité telle que définie sous-section 3.1.2-D. En effet, la fonction décrite sous-section 3.2.2.2 identifie les séquences de A partageant t k-mers avec les séquences de B indexées. Or, ceci est moins strict que trouver les séquences de A partageant au moins t k-mers avec *au moins une séquence* de B. En effet, quand on calcule $A \overrightarrow{\cap} B$, les t k-mers partagés de A peuvent être répartis sur plusieurs séquences de B. Autrement dit, un faux positif existe pour $t > 1$ quand les t k-mers partagés d'une séquence requête se trouvent sur au moins deux séquences du jeu indexé (voir figure 19). Ce type de faux positif est appelé **faux positif de séquence**. Même sans faux positifs dus au filtre de BLOOM, cet effet serait présent car la structure d'indexation ne conserve que la présence d'un k-mer dans un ensemble de séquence. Quand on indexe un k-mer dans le filtre de BLOOM, on perd l'information liant un k-mer et sa séquence d'origine : on ne peut savoir si deux k-mers distincts proviennent de la même séquence ou non.

3.2.3.3 *Diminution des faux positifs de séquence grâce aux trois étapes*

La figure 20 illustre comment les trois étapes consécutives permettent de retirer une partie des faux positifs de séquence. Dans l'exemple figure 20, on utilise $t = 2$. Seules les deux premières séquences des jeux de données A et B sont similaires et doivent être placées respectivement dans $(A \overrightarrow{\cap} B)$ et $(B \overrightarrow{\cap} A)$.

Lors de la première demi-intersection $(A \overrightarrow{\cap} B)^*$, ces deux séquences de A sont bien identifiées comme similaires. Les deux séquences suivantes ont chacune un seul k-mer partagé (en mauve puis en vert) avec le jeu B et ne sont donc pas placées dans le jeu résultat. La cinquième séquence partage deux k-mers (le rouge et le violet) avec le jeu de données B et est donc placées dans le résultat. De même, la dernière séquence partage deux k-mers avec B. Ces deux dernières séquences partagent des k-mers avec des séquences *distinctes* de B et sont donc des faux positifs de séquence. À la fin de cette première étape, la demi-intersection $(A \overrightarrow{\cap} B)^*$ contient les deux vraies séquences à retrouver mais aussi deux faux positifs de séquence.

La seconde étape consiste à calculer $B \overrightarrow{\cap} (A \overrightarrow{\cap} B)^*$. Les deux premières séquences de B sont bien identifiées comme similaires et placées dans le résultat $(B \overrightarrow{\cap} A)$. La troisième séquence ne partage aucun k-mer (en mauve et en vert) avec $(A \overrightarrow{\cap} B)^*$. Les deux dernières séquences partagent chacune un seul k-mer avec l'index et ne sont donc pas similaires. À la fin de la seconde étape, la demi-intersection $(B \overrightarrow{\cap} A)$ ne comporte que les vrais positifs de B.

La dernière étape consiste à calculer $(A \overrightarrow{\cap} B)^* \overrightarrow{\cap} (B \overrightarrow{\cap} A)$. Les deux premières séquences sont naturellement identifiées comme similaires et placées dans le résultat $(A \overrightarrow{\cap} B)$. La troisième séquence ne partage aucun k-mer avec le jeu $(B \overrightarrow{\cap} A)$ et n'est donc pas insérée dans le résultat. La dernière séquence partage deux k-mers avec $(B \overrightarrow{\cap} A)$ mais sur des séquences *distinctes* de B et est donc un faux positif. À la fin de cette dernière étape, la demi-intersection $(A \overrightarrow{\cap} B)$ contient les deux vraies séquences à retrouver et un faux positif de séquence.

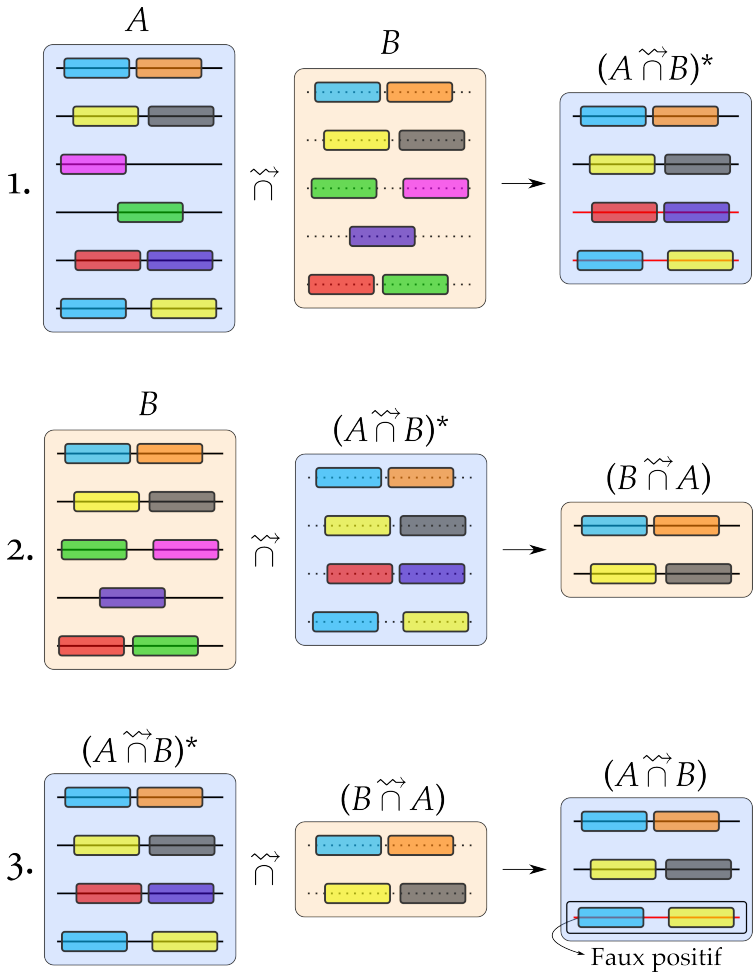


FIGURE 20: Réduction des faux positifs de séquence par le pipeline complet.

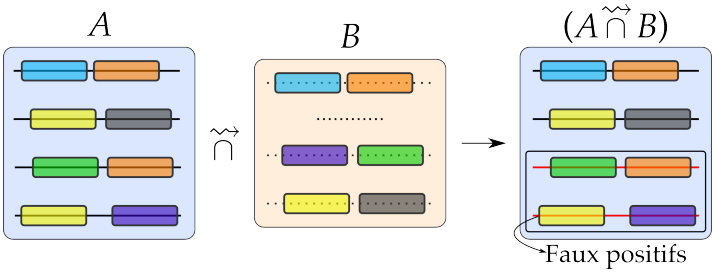
La seconde étape permet d'enlever les séquences non-similaires de B. Ces séquences enlevées créaient un faux positif sur le jeu A (séquence contenant le k-mer rouge et le k-mer violet). Mais, la dernière séquence de A reste un faux positif de séquence dans $(A \overrightarrow{\cap} B)$. Cette séquence contient des k-mers partagés avec des séquences distinctes de B, elles-mêmes présentes dans $(B \overrightarrow{\cap} A)$. Ainsi, même durant la troisième étape, ces k-mers sont partagés dans $(B \overrightarrow{\cap} A)$ et la séquence de A les contenant est donc placée dans $(A \overrightarrow{\cap} B)$.

Ainsi, calculer les jeux résultats en appliquant l'opération fondamentale $\overrightarrow{\cap}$ entre un jeu requête et un jeu indexé résultant déjà d'une opération $\overrightarrow{\cap}$ asymétrique permet de retirer une partie des faux positifs de séquence. Les faux positifs restants sont caractérisés par t k-mers partagés sur au moins deux séquences distinctes du jeu indexé, elles-mêmes considérées comme similaires à des séquences du jeu requête. Bien que cet effet soit difficile à quantifier, on montre section 4.2 que notre méthode permet d'obtenir des résultats fiables et très similaires à ceux obtenus par des méthodes plus classiques sur des jeux de données réels.

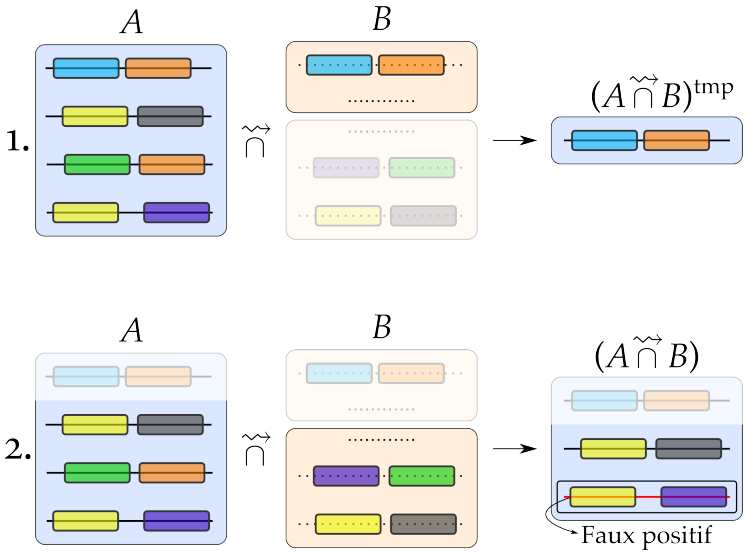
3.2.3.4 *Diminution des faux positifs de séquence par partitionnement de l'index*

Outre la diminution des collisions dans le filtre de Bloom (voir sous-section 3.2.2.2), le partitionnement de l'index permet aussi de retirer une partie des faux positifs de séquence, comme montré figure 21, en utilisant $t > 1$. Seules les deux premières séquences du jeu de données A sont similaires à des séquences de B et doivent être placées dans $(A \overrightarrow{\cap} B)$.

En traitant l'index en une seule fois, figure 21a, le calcul de la demi-intersection $(A \overrightarrow{\cap} B)$ conduit à trouver deux faux positifs de séquence, pour les mêmes raisons qu'expliquées par-



(a) Le calcul de $A \overset{\rightsquigarrow}{\cap} B$ sans découpage de l'index conduit à deux faux positifs de séquence.



(b) Le même calcul avec découpage de l'index enlève un des faux positifs de séquence.

FIGURE 21: Diminution des faux positifs de séquence par partitionnement de l'index. Les lignes en pointillés noir représentent plusieurs séquences non pertinentes pour l'exemple. Les parties plus claires ne sont pas utilisées.

tie 3.2.3.3.

Dans l'exemple présenté figure 21b, en divisant l'index en deux parties le calcul de $(A \overset{\rightsquigarrow}{\cap} B)$ s'effectue en deux temps. Tout d'abord, les premières séquences du jeu B sont utilisées comme index. Lors de cette étape, seule la première séquence de A (avec le k-mer bleu et le k-mer orange) est retrouvée et stockée dans le résultat intermédiaire $(A \overset{\rightsquigarrow}{\cap} B)^{tmp}$. Lors de la seconde étape, on indexe le reste de B. Seules les séquences de A n'étant pas déjà présentes dans $(A \overset{\rightsquigarrow}{\cap} B)^{tmp}$ sont traitées, la première séquence est donc ignorée. La séquence contenant les k-mers jaune et gris est retrouvée et placée dans le jeu résultat. L'avant-dernière séquence (k-mer vert et k-mer orange) n'est pas retrouvée car elle ne partage qu'un seul k-mer (le vert) avec l'index. Finalement, la dernière séquence (k-mer jaune et k-mer violet) est retrouvée dans la partie indexée. À la fin de cette dernière étape, la demi-intersection $(A \overset{\rightsquigarrow}{\cap} B)$ contient les deux vraies séquences à retrouver et un faux positif de séquence. Le faux positif qui a été enlevé partageait ses k-mers avec deux séquences distinctes qui n'ont pas été indexées en même temps.

Ainsi, calculer les jeux résultats en appliquant l'opération fondamentale $\overset{\rightsquigarrow}{\cap}$ entre un jeu requête et un jeu indexé divisé en plusieurs parties permet de retirer une partie des faux positifs de séquence. Les faux positifs restants sont caractérisés par t k-mers partagés sur au moins deux séquences distinctes du jeu indexé et indexées **simultanément**. Comme pour la diminution des faux positifs de séquence grâce aux trois étapes de notre méthode, la réduction de ces faux positifs est difficile à quantifier. Le partitionnement de l'index ralenti le processus : certaines séquences requêtes sont testées plusieurs fois, comme les quatre séquences non grisées de A figure 21b (2). Toutes les séquences non-similaires de A sont ainsi testées pour chaque partition de l'index, au lieu d'une seule fois si l'index est traité en une seule étape.

3.2.3.5 *Diminution des faux positifs du filtre de* Bloom *grâce aux trois étapes et au partitionnement de l'index*

Combiner les trois étapes du pipeline complet et la division des index permet de retirer une partie des faux positifs du filtre de Bloom. En effet, lors de la première étape, certains k-mers du jeu A sont retrouvés à tord dans un des index de B avec une probabilité p. Or, lors de la troisième étape, ces mêmes k-mers vont être à nouveau testés, mais contre un index différent : seules les séquences similaires de B sont alors indexées. Par exemple, les n premières premières séquences de B sont utilisées lors du premier index de la première étape. Lors de la troisième étape, une partie x de ces n séquences n'est plus présente dans $(B \overset{\rightsquigarrow}{\cap} A)$ et est donc remplacée par les x séquences similaires suivantes de B. La probabilité qu'un k-mer soit alors un faux positif à la fois dans B et dans $(B \overset{\rightsquigarrow}{\cap} A)$ est difficile à évaluer car liée au nombre x de nouvelles séquences.

Il est par contre possible de borner cette probabilité. Si l'index est exactement le même lors de la troisième étape (c'est à dire que toutes les x premières séquences de B sont similaires), les probabilités qu'un k-mer soit un faux positif dans B et qu'un k-mer soit un faux positif à la fois dans B et dans $(B \overset{\rightsquigarrow}{\cap} A)$ sont identiques. À l'inverse, si l'index est totalement différent (c'est à dire qu'aucune des x premières séquences n'a été trouvée lors de la seconde étape), les deux index sont indépendants. Donc, la probabilité de trouver un faux positif lors de la troisième étape est p et dès lors, la probabilité qu'un k-mer soit un faux positif lors de la première étape **et** lors de la dernière étape est p^2.

Concernant le résultat $(A \overset{\rightsquigarrow}{\cap} B)$, la probabilité qu'un k-mer soit un faux positif du filtre de Bloom est donc dans $[p^2, p]$, avec $p = 0{,}1\%$ pour les paramètres standards (voir sous-section 3.3.1.4). Ce n'est par contre pas le cas pour $(B \overset{\rightsquigarrow}{\cap} A)$ car seule une étape est réalisée pour cette

demi-intersection ; la probabilité est alors p.

Il est important de noter que les séquences non similaires de la première étape ne doivent pas être ré-utilisées lors de la troisième étape. Notre méthode ne générant pas de faux négatifs, une séquence non similaire dans une des étapes est éliminée à raison. Si on la réutilise lors des étapes suivantes, elle peut causer des faux positifs dans le filtre de Bloom, voire devenir un faux positif de séquence. Par exemple, prenons une séquence partageant exactement $t-1$ k-mers avec l'index lors de la première étape. Cette séquence ne doit pas être retenue car elle ne rentre pas dans les critères de séquence similaire. Si on la garde, elle peut générer des faux positifs dans le filtre de Bloom lors de la seconde étape. Pire, il est possible que lors de la dernière étape, elle partage toujours ses vrais $t-1$ k-mers avec l'index et qu'un autre de ses k-mers soit un faux positif. Cette séquence sera alors considérée comme présente dans le jeu indexé, quand bien même elle était identifiée comme non similaire lors de la première étape. Ainsi, durant toutes les étapes, une séquence non similaire est systématiquement ignorée pendant le reste du traitement.

CONCLUSION Notre méthode se base sur une notion de similarité approximative, plus simple qu'un d'alignement de séquence. La similarité approximative exprimée entre deux jeux de données conduit à l'obtention de deux score de similarité. Le premier est lié à un échantillon, *i.e.* cet échantillon à tel score de similarité comparé à cet autre échantillon. Le second score de similarité est global et symétrique pour deux échantillons, *i.e.* ces deux échantillons on tel score de similarité entre eux.

Ces deux scores de similarité nécessitent de réaliser des intersections de jeux de données. La mise en place de ces intersections passe par plusieurs phases d'indexation et de requête dans un filtre de Bloom. À la fois le filtre de Bloom mais aussi la méthode en soit conduisent à la génération de faux positifs. Une partie de ces faux positifs est éliminée par notre méthodologie.

Dans la section suivante, la mise en œuvre de notre méthodologie est détaillée, entre autre au niveau de la structure de données utilisée.

3.3 MISE EN ŒUVRE

La mise en œuvre de notre méthodologie présentée précédemment (voir section 3.2) nécessite l'indexation d'une grande quantité de k-mers. En effet, l'opération fondamentale de notre méthode consiste à tester si un k-mer donné est présent dans un ensemble de k-mer indexés. Un jeu de données métagénomiques issu de NGS peut contenir 100 millions de séquences de 100 pb, soit 6,8 milliards de k-mers pour $k = 33$. Pour comparer toutes les séquences, l'opération fondamentale est ainsi menée plusieurs milliards de fois : la structure d'indexation doit donc exécuter cette opération le plus rapidement possible. De plus, la quantité de données actuellement générées dans le cadre de projets de métagénomique oblige à utiliser des structures d'indexation très légère en mémoire.

Comme expliqué dans l'état de l'art (voir sous-section 2.3.5), le filtre de Bloom est très rapide, tant à la création de l'index qu'à l'interrogation de l'index. De plus cette structure est très légère en mémoire. Dans la sous-section 3.3.1, nous proposons une structure d'indexation basée sur le Bloom, donc qui n'est pas excessive en mémoire, mais façonnée de manière à être plus rapide d'un ordre de grandeur qu'un filtre de Bloom conventionnel. Cette structure est cependant, comme le filtre de Bloom, sujette aux faux positifs.

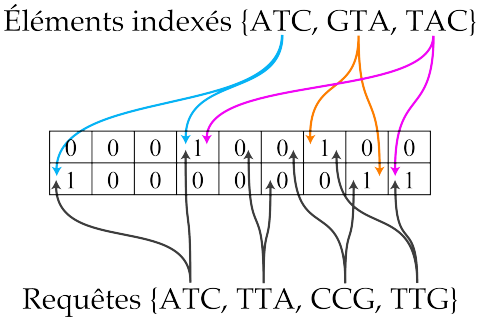


FIGURE 22: Représentation du BDS pour 2 fonctions de hachage. Chaque fonction de hachage utilise un tableau de bits qui lui est dédié. L'élément TTG est retrouvé à tord : c'est un faux positif.

Notre méthodologie a été implémentée, en utilisant cette structure de données, dans un outil dédié à la métagénomique comparative *de novo* : COMPAREADS. Cet outil est présenté sous-section 3.3.2.

3.3.1 Une variation du filtre de Bloom : le BDS

Dans cette section, nous détaillons une variation du filtre de Bloom. Contrairement au filtre de Bloom traditionnel qui utilise un unique tableau de bits, dans notre structure chaque fonction de hachage utilise un tableau de bits qui lui est dédié et donc distinct des autres fonctions de hachage (voir figure 22). En terme de faux positifs, il a été montré que cette spécificité est asymptotiquement équivalente à la définition initiale du filtre de Bloom [135]. Par contre, cette définition permet de calculer et d'écrire en parallèle les codes de hachage d'un élément à insérer sans risque d'accès concurrent à un même bit. En plus de cette particularité, un jeu de fonctions de hachage particulier a été défini. Pour éviter toute confusion avec le filtre de Bloom classique, cette structure de données sera dénommée BDS dans le reste du document (pour *Bloom Data Structure*).

Le BDS profite des avantages du filtre de Bloom : il est très efficace à l'indexation et à la recherche, et son occupation mémoire ne dépend pas de la quantité de données à indexer. Par contre, il est également sujet aux faux positifs. Comparé au filtre de Bloom classique, le BDS à l'avantage d'utiliser des fonctions de hachage très rapides à calculer. Insérer un élément ou tester la présence d'un élément dans le filtre se fait un ordre de grandeur plus rapidement qu'avec les fonctions de hachage traditionnellement utilisées (voir 4.1.2). Cette rapidité se fait au détriment de l'utilisation mémoire et le BDS nécessite ainsi 2,5 fois plus d'espace qu'un filtre de Bloom classique.

3.3.1.1 *Définition de nouvelles fonctions de hachage*

Le BDS est utilisé pour indexer tous les k-mers apparaissant au moins une fois dans un jeu de données. Les fonctions de hachage utilisées dans le BDS forment une famille particulière. Ces fonctions sont calculées très efficacement sur les k-mers consécutifs, car le calcul du code de hachage d'un k-mer apparaissant en position i est fait en temps constant dès lors que le code de hachage du k-mer apparaissant en $i-1$ est déjà calculé. En effet, seul le nucléotide différent entre ces deux k-mers est alors haché.

DÉFINITION DE LA FAMILLE DE FONCTIONS UTILISÉES Les fonctions de hachage définies ici attribuent à un k-mer une séquence de k bits. Chaque nucléotide est associé à un bit à 0 ou à 1 et cette association ne dépend que du type du nucléotide (A, C, G ou T). Les sept fonctions définies équations 8 et 9 répondent à ces critères. Il existe d'autres fonctions basées

sur les mêmes critères, mais elles apportent toute une information redondante à l'une des sept fonctions définies ici. On peut catégoriser ces sept fonctions en deux groupes. Le premier groupe (équation 8) contient trois fonctions dites "équilibrées". Le second groupe (équation 9) contient les quatre autres fonctions, les fonctions "non-équilibrées".

$$f_j : \Sigma^k \to \{0,1\}^k : \forall i \in [0,k-1] \begin{cases} f_1(s)[i] = 0 & \text{si} \quad s[i] = A \text{ ou } C \qquad f_1(s)[i] = 1 \quad \text{sinon} \\ f_2(s)[i] = 0 & \text{si} \quad s[i] = A \text{ ou } G \qquad f_2(s)[i] = 1 \quad \text{sinon} \\ f_3(s)[i] = 0 & \text{si} \quad s[i] = A \text{ ou } T \qquad f_3(s)[i] = 1 \quad \text{sinon} \end{cases} \tag{8}$$

$$f_j : \Sigma^k \to \{0,1\}^k : \forall i \in [0,k-1] \begin{cases} f_4(s)[i] = 0 & \text{si} \quad s[i] = A \qquad f_4(s)[i] = 1 \quad \text{sinon} \\ f_5(s)[i] = 0 & \text{si} \quad s[i] = C \qquad f_5(s)[i] = 1 \quad \text{sinon} \\ f_6(s)[i] = 0 & \text{si} \quad s[i] = G \qquad f_6(s)[i] = 1 \quad \text{sinon} \\ f_7(s)[i] = 0 & \text{si} \quad s[i] = T \qquad f_7(s)[i] = 1 \quad \text{sinon} \end{cases} \tag{9}$$

A. FONCTIONS ÉQUILIBRÉES. Pour les fonctions f_1, f_2 et f_3 (équation 8), la valeur 0 ou 1 peut provenir d'exactement deux nucléotides distincts. Ainsi, en supposant une composition uniforme en nucléotides d'un k-mer, les trois codes de hachage de ces fonctions sont composés d'autant de 0 que de 1.

B. FONCTIONS NON-ÉQUILIBRÉES. Pour les fonctions f_4, f_5, f_6 et f_7 (équation 9), la valeur 0 provient d'un nucléotide donné mais la valeur 1 peut provenir de trois nucléotides différents. Ainsi, en supposant une composition uniforme en nucléotides d'un k-mer, les quatre codes de hachage de ces fonctions comportent trois fois plus de 1 que de 0.

PROPRIÉTÉ ALGORITHMIQUE Ces sept fonctions permettent d'avoir une relation très simple entre les codes de hachage de deux k-mers consécutifs. Le code de hachage d'un k-mer peut être rapidement calculé en partant du code de hachage du précédent k-mer. Il suffit d'effectuer un décalage à gauche de la séquence binaire correspondant au k-mer précédent et de calculer seulement le code de hachage du nouveau nucléotide à insérer à la fin de la nouvelle séquence (voir tableau 4). Il est important de noter que cette propriété permet de calculer le code de hachage d'un nucléotide donné une seule et unique fois par séquence, lorsque tous les k-mers de cette séquence sont indexés : pour chaque position dans une séquence, le nucléotide qui y est présent est traité une seule fois, évitant ainsi de répéter plusieurs fois le même calcul.

Ces fonctions ne sont pas des fonctions de hachage classiques, mais elles présentent de bonnes propriétés. Dans la section 4.1.2, les performances de ces fonctions sont comparées avec des fonctions de hachage classiques en terme de temps de calcul et de taux de faux positifs.

3.3.1.2 *Étude des faux positifs du* BDS

Le BDS étant une extension du filtre de Bloom, il est aussi sujet aux faux positifs, c'est à dire qu'il peut considérer à tord qu'un k-mer est indexé dans la structure (voir sous-section 2.3.5). Dans cette sous-section, on analyse la variation du taux de faux positifs pour chaque fonction de hachage et leurs combinaisons, en fonction de la taille de k et du nombre n de k-mers distincts indexés. Les probabilités de faux positifs sont présentées équation 10 pour les fonctions de hachage équilibrées et équation 11 pour les non-équilibrées.

T	A	C
1	0	0

$f_1(TAC) = 100$

(a) Premier 3-mer : tous les bits doivent être calculés (en vert).

T	A	C	G
1	0	0	1

$f_1(ACG) = 001$

(b) Second 3-mer : seul le dernier bit (en vert) doit être calculé. Le premier bit (en orange) est purgé par décalage à gauche de la séquence binaire.

A	C	G	A
0	0	1	0

$f_1(CGA) = 010$

(c) Dernier 3-mer : comme précédemment, seul le dernier bit (en vert) doit être calculé et le premier bit (en orange), purgé.

TABLE 4: Exemple de calcul des codes de hachage des 3-mers de la séquence TACGA par la fonction de hachage f_1

CAS CLASSIQUE DE COLLISIONS DE FONCTIONS DE HACHAGE Une fonction de hachage transforme un ensemble de départ en un ensemble d'arrivée. Une "collision" d'une fonction de hachage correspond à au moins deux données distinctes de l'ensemble de départ qui génèrent le même code de hachage. Il est alors impossible de savoir à quelle donnée correspond le code de hachage obtenu. On appelle fonction de hachage "parfaite" une fonction qui ne génère aucune collision. En pratique, il est très coûteux d'utiliser une fonction de hachage parfaite [135].

DES COLLISIONS PAR FONCTION DE HACHAGE Chaque fonction de hachage définie sous-section 3.3.1.1 génère des collisions. Par exemple, en utilisant uniquement la fonction f_1 ($A/C \rightarrow 0$ $G/T \rightarrow 1$), il est impossible de savoir si le code de hachage 00 correspond au mot AA, CC, AC OU CA.

COMBINAISON DE FONCTION DE HACHAGE Il est important de remarquer qu'en combinant plusieurs fonctions de hachage, il devient possible d'identifier de manière certaine chaque élément. Ainsi, deux fonctions équilibrées apportent suffisamment d'information pour pouvoir discriminer deux séquence d'ADN, quelles qu'elles soient. De même, trois fonctions non-équilibrées sont aussi suffisantes. Autrement dit, pour $i,j \in \{1,2,3\}$, il n'existe pas deux mots w_1,w_2 tels que $f_i(w_1) = f_i(w_2)$ et $f_j(w_1) = f_j(w_2)$. De même, pour $i,j,k \in \{4,5,6,7\}$, il n'existe pas deux mots w_1,w_2 tels que $f_i(w_1) = f_i(w_2)$, $f_j(w_1) = f_j(w_2)$ et $f_k(w_1) = f_k(w_2)$. Ainsi, un k-mer correspond à un **unique** ensemble de bits dans le filtre, que l'on nommera "empreinte".

FAUX POSITIFS DU BDS L'utilisation de plusieurs fonctions de hachage permet d'avoir une empreinte unique par k-mer dans le BDS. Mais, une empreinte unique ne signifie pas aucun faux positif. En effet, il est possible de trouver dans le filtre un ensemble de bits correspondant à un k-mer qui n'y est pas indexé. Les bits d'une empreinte ne sont pas liés les uns aux autres, il est impossible de savoir si deux bits codent pour le même k-mer ou non. Il est alors possible, quand plusieurs empreintes sont présentes dans le filtre en même temps, de trouver d'autres empreintes qui ne correspondent pas à des k-mers indexés (voir tableau 5). C'est ce type d'erreurs que l'on appelle faux positifs du BDS.

A. PROBABILITÉ DE FAUX POSITIFS PAR FONCTION. Considérant que la composition en nucléotides des k-mers indexés est uniforme, il est possible de calculer la probabilité $P_{FP}(f_i,k,n)$,

111				
110		*		**
101	*	*	*	*
100	*	*		
011				
010			*	
001			*	
000	*			
	f_1	f_2	f_3	f_4

(a) Les trois k-mers TAG, TCA et CAC sont indexés dans le BDS. Chaque k-mer a une empreinte unique, mais il y a une collision entre TCA et CAC sur la fonction f_4.

111				
110		*		**
101	*	*	*	*
100	*	*		
011				
010			*	
001			*	
000	*			
	f_1	f_2	f_3	f_4

(b) En l'absence de lien entre un k-mer et ses codes de hachage, il est possible de croire à tord que le k-mer TAC est indexé. C'est un faux positif.

TABLE 5: Exemple de collision de k-mers (5a) et exemple de faux positif (5b). Seuls les trois k-mers TAG, TCA et CAC sont indexés dans le BDS, mais il est tout de même possible d'y trouver le k-mer TAC.

où k est la taille du mot et n le nombre de mots distincts indexés, pour n'importe quelle requête, d'être un faux positif avec l'une des sept fonctions de hachage f_i. Cette probabilité dépend du nombre de k-mers distincts ayant le même code de hachage pour cette fonction. On notera que pour les fonctions équilibrées f_1, f_2 et f_3, la valeur 0 ou 1 est obtenue à partir de deux nucléotides différents. Ainsi, le nombre de k-mers partageant le même code de hachage est le même pour tous les k-mers, et est égal à 2^k. La probabilité que deux k-mers aient un code de hachage différent est alors $1 - \frac{2^k}{4^k} = 1 - \frac{1}{2^k}$, et la probabilité d'avoir au moins un k-mer parmi les n indexés ayant un code de hachage partagé est donc :

$$\forall i \in \{1,2,3\} P_{FP}(f_i,k,n) = 1 - (1 - \frac{1}{2^k})^n \tag{10}$$

Il est intéressant de noter que cela correspond à la probabilité de faux positif de n'importe quelle fonction de hachage qui distribue les codes de hachage uniformément dans un tableau de 2^k bits, telles les fonctions de Jenkins utilisées en comparaison sous-section 4.1.2.

Pour les fonctions non-équilibrées, la valeur 0 code pour seulement un nucléotide parmi {A, C, G, T}. Le nombre de k-mers partageant le même code de hachage dépend donc du nombre de 0 du code de hachage de la requête. Pour un k-mer donné ayant x 0 dans son code de hachage, la probabilité ci-dessus devient pour les fonction f_4, f_5, f_6 et f_7 : $1-(1-(\frac{1}{4})^x(\frac{3}{4})^{(k-x)})^n$. Pour obtenir la probabilité pour n'importe quel k-mer, il faut sommer les différentes valeurs de x de cette probabilité, pondérée par la probabilité que le code de hachage d'un k-mer contienne x 0. La composition d'un nucléotide donné dans

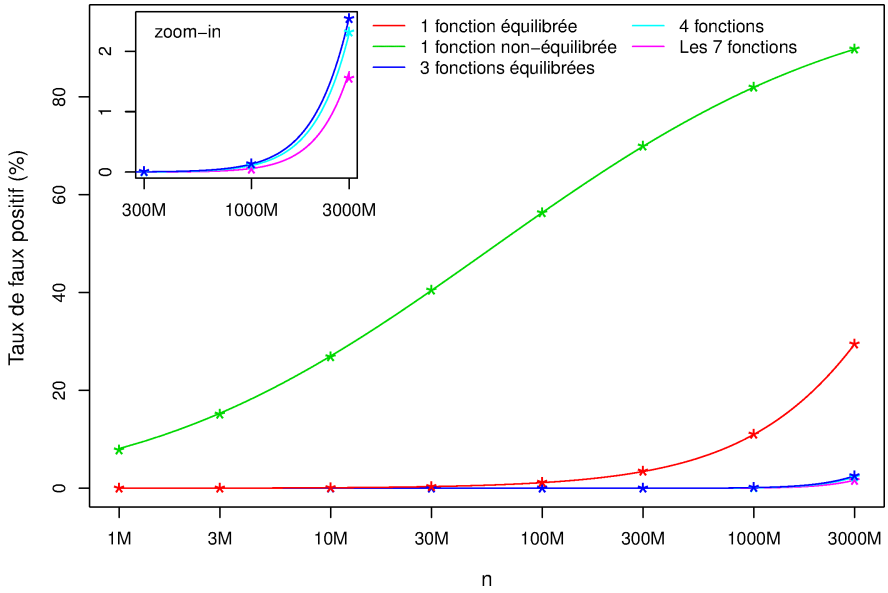


FIGURE 23: Taux de faux positifs pour différentes combinaisons de fonctions de hachage en fonction du nombre de k-mers indexés (échelle logarithmique). Les lignes correspondent aux prédictions théoriques et les points aux observations empiriques obtenues par simulation. Les données ont été obtenues pour $k = 33$. La combinaison "4 fonctions" est composée des trois fonctions équilibrées et d'une fonction non-équilibrée.

un k-mer de longueur k, considérant une composition uniforme, suit une distribution binomiale, et on obtient alors :

$$\forall i \in \{4,5,6,7\} P_{FP}(f_i,k,n) = \sum_{x=0}^{k} \binom{k}{x} a_x (1-(1-a_x)^n)$$
$$\text{avec } a_x = (\frac{1}{4})^x (\frac{3}{4})^{k-x} \tag{11}$$
$$\text{et où } \binom{k}{x} \text{ est le coefficient binomial, } i.e. \binom{k}{x} = \frac{k!}{x!(k-x)!}$$

On peut voir figure 23 que les fonctions équilibrées apportent beaucoup moins de faux positifs que les non-équilibrées. Ceci s'explique car pour une fonction non-équilibrée, étant donné un k-mer avec une composition "normale" et donc 25% de 0 dans son code de hachage, il y a beaucoup plus de k-mers ayant le même code de hachage que pour les fonctions équilibrées, car : $3^{\frac{3k}{4}} \gg 2^k$.

B. PROBABILITÉ DE FAUX POSITIF PAR COMBINAISON DE FONCTIONS. Lorsque l'on combine plusieurs fonctions de hachage dans le BDS, obtenir un faux positif sur une requête signifie que toutes les fonctions retournent un faux positif. On a déjà remarqué qu'il n'existe pas deux k-mers distincts qui ont le même couple de code de hachage pour deux fonctions équilibrées, peu importent lesquelles. Ceci implique que la probabilité d'avoir un faux positif sur une fonction ne dépend pas du résultat d'une autre fonction, mis à part le fait que le nombre de k-mers indexés qui peuvent être des faux positifs est alors plus faible (n dans l'équation 10) : en effet, si x k-mers partagent le même code de hachage pour une fonction donnée, ces k-mers ont une probabilité nulle d'avoir le même code de hachage pour une autre fonction. Il faut noter que cet effet est négligeable car n est toujours très grand. Ainsi, le produit des probabilités individuelles de chaque fonction équilibrée donne le maximum théorique de faux positifs suivant :

$$P_{FP}(f_1 \cap f_2 \cap f_3,k,n) \lesssim (1-(1-\frac{1}{2^k})^n)^3 \tag{12}$$

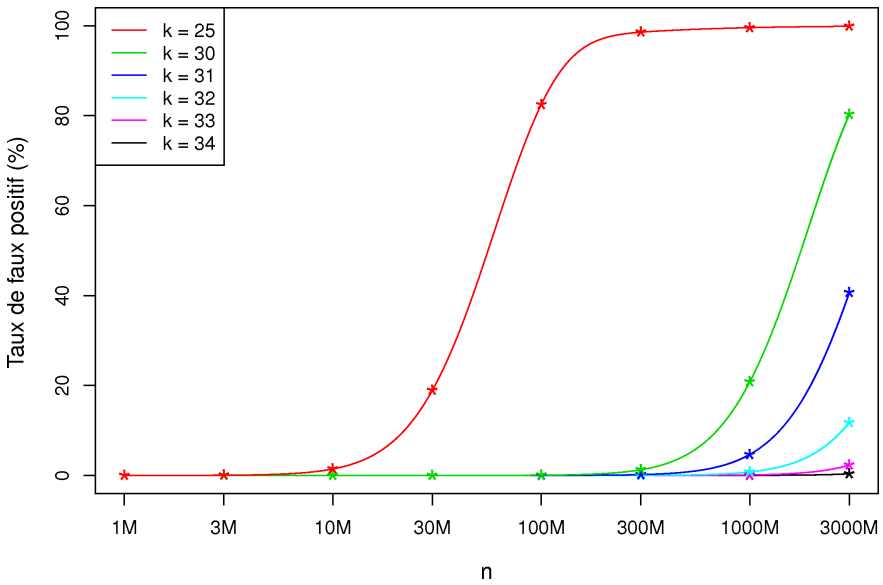


FIGURE 24: Taux de faux positifs pour différentes valeurs de k en fonction du nombre de k-mers indexés (échelle logarithmique). Les lignes correspondent aux prédictions théoriques et les points aux observations empiriques obtenues par simulation. Les données ont été obtenues pour la combinaison de quatre fonctions de hachage, les trois fonctions équilibrées et une fonction non-équilibrée.

Cette propriété d'indépendance implique que combiner ces trois fonctions dans le BDS est une stratégie très efficace pour réduire les faux positifs, comme on peut le voir figure 23, surtout pour de grandes valeurs de n.

Concernant les fonctions non-équilibrées, cette propriété d'indépendance est perdue. Il est en effet possible de trouver deux k-mers distincts qui ont les mêmes codes de hachage pour au moins deux fonctions non-équilibrées, ou une fonction équilibrée et au moins une fonction non-équilibrée. Par exemple, les 2-mers TG et TT ont le même code de hachage, 11, que ce soit pour les fonctions de hachage f_1, f_4 ou f_5. Le taux de faux positifs théorique des fonctions non-équilibrées est alors bien plus difficile à calculer. Des simulations ont permis de calculer un taux de faux positif empirique (voir figure 23). Il apparait que ces résultats empiriques sont très proches de la formule obtenue par la multiplication des différentes probabilités individuelles, donc en supposant malgré tout une complète indépendance entre toutes les fonctions de hachage.

3.3.1.3 *Compromis entre temps d'exécution et empreinte mémoire*

Le choix du paramètre k ainsi que des fonctions à utiliser doivent être déterminés afin d'obtenir le meilleur compromis entre l'utilisation mémoire et le taux de faux positifs. La comparaison des courbes figure 23 conduit à utiliser la combinaison "4 fonctions", contenant les trois fonctions équilibrées et une fonction non-équilibrée. En effet, les fonctions non-équilibrées ne sont pas essentielles car elles ont un impact limité sur le taux de faux positifs. De plus, les sept fonctions de hachage ont toutes une occupation mémoire identique et liée à la taille k des mots à indexer, mais pas à la quantité de mots à indexer. La quantité de mémoire utilisée par le BDS est donc dépendante de k et du nombre de fonctions de hachage utilisées. Pour rappel, chaque fonction de hachage est associée à un unique tableau de bits utilisant 2^k bits. En utilisant les sept fonctions de hachage, le BDS a une empreinte mémoire totale de $7 \cdot 2^k$ bits et peut être stocké sur 2^k octets. En utilisant seulement quatre fonctions, le BDS occupe $4 \cdot 2^k$

bits et peut alors être stocké sur deux fois moins d'espace, 2^{k-1} octets.

Pour cette combinaison de quatre fonctions de hachage, la figure 24 montre le taux de faux positifs en fonction du nombre n de mots distinct indexés, obtenu pour plusieurs valeurs de k. Plus k est grand, plus le taux de faux positifs est faible pour la même valeur n. Ainsi, plus k est grand, plus on peut indexer de k-mers tout en maintenant un taux de faux positifs sous un seuil. D'un autre côté, l'utilisation mémoire est en $O(2^k)$. On peut voir figure 24 qu'utiliser une taille de k-mers d'au moins 30 permet d'indexer au moins 300 millions de k-mers pour un taux de faux positifs de moins de 2%. Biologiquement, il a été montré qu'une sous-séquence de taille $k \geqq 20$ tend à apparaitre de manière unique dans un génome [137, 138].

Pour $k = 33$, on peut indexer 1 milliard de k-mers et obtenir un taux théorique de faux positifs de 0,13% (basé sur l'équation 12 pour les trois fonctions équilibrées). En utilisant les quatre fonctions choisies, le taux de faux positifs est encore plus faible ; il a été estimé empiriquement à 0,114% (voir figure 23). Ainsi, utiliser quatre fonctions de hachage (toutes les fonctions équilibrées et une non-équilibrée) et un k à 33 est un bon jeu de paramètres pour indexer 1 milliard de k-mers distincts. Avec ces paramètres, l'impact mémoire du BDS est de 4 Go, et donc utilisable sur les ordinateurs standards.

3.3.1.4 *Limitation de l'espace utilisé par l'indexation*

Comme pour le filtre de Bloom classique (voir sous-section 3.3.1.2), le taux de faux positifs augmente avec le remplissage du BDS. Afin de limiter et contrôler ce taux, la phase d'indexation est interrompue quand le volume de k-mers des premières séquences de B dépasse un seuil l (voir sous-section 3.2.2.2). Ce seuil peut être calculé à partir de l'équation 12 afin de maintenir un taux de faux positifs de 0,1%, peut importe le k utilisé. On obtient alors l'équation suivante :

$$l = \frac{\log(0{,}9)}{\log(1 - 0{,}5^k)} \tag{13}$$

Comme l'équation 12 permet de calculer le taux de faux positifs pour la combinaison des trois fonctions balancées, le taux effectif de faux positifs du BDS, qui utilise quatre fonctions, est encore inférieur. Ainsi, pour $k = 33$, l'équation 13 permet de définir un seuil à $9{,}1 \times 10^8$ k-mers à indexer au maximum. Empiriquement, nous avons constaté qu'il est possible d'indexer un milliard de 33-mers pour un taux de faux positifs de 0,114% (voir figure 23). Il est intéressant de noter qu'incrémenter de 1 la valeur de k conduit à la fois à doubler la mémoire nécessaire mais aussi à doubler la quantité de k-mers qu'on peut indexer pour le même taux de faux positif de 0,1%.

CONCLUSION SUR LE BDS Cette nouvelle structure, basée sur le filtre de Bloom, a une empreinte mémoire qui n'est pas liée à la quantité de k-mers à indexer, mais à la taille k de ces k-mers. La quantité de séquences présentes dans un jeu de données n'a donc pas d'impact sur l'utilisation mémoire du BDS. De plus, cette structure utilise des fonctions de hachage très rapides : pour calculer tous les k-mers d'une séquence, chaque position dans la séquence n'est hachée qu'une seule et unique fois. Cette structure a par contre le désavantage, comme le filtre de Bloom, de générer des faux positifs.

Le nombre de faux positifs est lié au remplissage du BDS. La quantité de mémoire utilisée par le BDS n'est pas dépendante du nombre d'éléments à indexer, mais il est important de définir un bon compromis entre nombre d'éléments à indexer, quantité de mémoire à utiliser

et taux de faux positifs voulu. Dans notre cas, on accepte d'avoir un taux de faux positif d'environ 0,1%. Dès lors on peut indexer 1 milliard de 33-mers en utilisant 4 Go de mémoire.

3.3.2 Un nouvel outil : CompareADS

La méthodologie présentée section 3.2 a été implémentée dans un outil nommé CompareADS. Dans ce qui suit, quelques détails techniques et pratiques sont expliqués.

3.3.2.1 *Implémentation et disponibilité*

Les deux fonctions de base de l'algorithme 4 sont Indexation (voir algorithme 1), qui indexe tous les k-mers d'un jeu de données dans le BDS, et Requête (voir algorithme 2), qui teste si les séquences d'un jeu de données sont similaires à une séquence de l'index. La fonction d'indexation d'une séquence et la fonction qui teste si une séquence est dans l'index sont détaillées ici plus précisément. Ces deux opérations sont très semblable et sont répétées pour chaque séquences ; elles sont présentées respectivement algorithme 5 et algorithme 6.

```
Données : Une séquence 'seq', la taille 'k' des k-mers, le tableau de bits 'BDS'

début
    pour i=0 a |seq|-1 faire
        décallage à gauche de clefA ;          /* Ajoute un zéro à la fin de clefA */
        si seq[i] == G ou seq[i] == T alors
            clefA |= 1
        fin

        décallage à gauche de clefB ;          /* Ajoute un zéro à la fin de clefB */
        si seq[i] == C ou seq[i] == T alors
            clefB |= 1
        fin

        décallage à gauche de clefC ;          /* Ajoute un zéro à la fin de clefC */
        si seq[i] == C ou seq[i] == G alors
            clefC |= 1
        fin

        décallage à gauche de clefD ;          /* Ajoute un zéro à la fin de clefD */
        si seq[i] == C ou seq[i] == G ou seq[i] == T alors
            clefD |= 1
        fin

        si i > k alors
            BDS[clefA] = 1
            BDS[clefB] = 1
            BDS[clefC] = 1
            BDS[clefD] = 1
        fin
    fin
fin
```

Algorithme 5: Fonction Indexer(seq) de CompareADS.

```
Données : Une séquence 'seq', la taille 'k' des k-mers, le nombre 't' de k-mers partagés
          pour qu'une séquence soit similaire, le tableau de bits 'BDS'
Résultat : Retourne 'vrai' si 'seq' partage au moins 't' k-mers avec le BDS, 'faux' sinon.

début
    tailleKmerCourant = 0
    pour i=0 a |seq|-1 faire
        décallage à gauche de clefA ;          /* Ajoute un zéro à la fin de clefA */
        si seq[i] == G ou seq[i] == T alors
            clefA |= 1
        fin

        décallage à gauche de clefB ;          /* Ajoute un zéro à la fin de clefB */
        si seq[i] == C ou seq[i] == T alors
            clefB |= 1
        fin

        décallage à gauche de clefC ;          /* Ajoute un zéro à la fin de clefC */
        si seq[i] == C ou seq[i] == G alors
            clefC |= 1
        fin

        décallage à gauche de clefD ;          /* Ajoute un zéro à la fin de clefD */
        si seq[i] == C ou seq[i] == G ou seq[i] == T alors
            clefD |= 1
        fin
        tailleKmerCourant++

        si tailleKmerCourant >= k alors
            si BDS[clefA] = 1 et BDS[clefB] = 1 et BDS[clefC] = 1 et BDS[clefD] = 1 alors
                nbKmersPartagés += 1
                si nbKmersPartagés >= t alors retourner vrai
                /* Remise à zéro des clefs                                          */
                clefA = 0
                clefB = 0
                clefC = 0
                clefD = 0
                tailleKmerCourant = 0
            fin
        fin
    fin

    retourner faux
fin
```

Algorithme 6: Fonction EstDansIndex(seq) de COMPAREADS.

COMPAREADS est une implémentation mono-cœur, en C, de notre méthodologie. Il utilise la combinaison de quatre fonctions de hachage présentée sous-section 3.3.1.3. Avec $k = 33$ et $t = 2$, le logiciel nécessite 4 Go de mémoire vive et peut donc fonctionner sur des ordinateurs standards. Avec les paramètres standards ($k = 33$, $t = 2$), COMPAREADS traite deux fichiers de chacun 100 millions de séquences de 100 pb en une dizaine d'heure, sur un cœur fonctionnant à 2,26 GHz.

Seul le paramètre k influe sur la taille de la mémoire, la quantité de données à traiter ne rentre pas en compte. Il est possible d'utiliser COMPAREADS sur des ordinateurs disposant d'une faible quantité de mémoire. Par exemple, en utilisant $k = 25$, la consommation mémoire est de 16 Mo. Pour maintenir un taux de faux positif raisonnable, COMPAREADS n'indexe alors pas plus de 3,5 millions de k-mers dans le BDS (voir équation 13). Pour indexer tous les 25-mers de 100 millions de séquences de 100 pb, on a alors besoin de 2172 BDS. Pour rappel, la complexité en temps de COMPAREADS est dépendante du nombre de BDS qu'on doit utiliser pour indexer l'ensemble d'un jeu de données (voir sous-section 3.2.2.2). Dès lors, comparer deux jeux de données de 100 millions de séquences en utilisant $k = 25$ n'est pas réalisable en quelques dizaines d'heures. Par contre, en comparant des jeux de données bien plus petits, par exemple 200 000 séquence de 100 pb par jeu, avec $k = 25$, l'ensemble des k-mers peut être indexé sur 5 BDS. COMPAREADS est alors en mesure de calculer efficacement l'intersection en utilisant 16 Mo de mémoire.

Si utiliser un k inférieur au paramètre standard peut être utile pour de petits jeux de données et sur des ordinateurs à faible mémoire, le lien direct entre k et la mémoire est le défaut majeur de notre méthode. Actuellement, il est presque impossible en pratique d'utiliser COMPAREADS avec une valeur de k à plus de 40. En effet, $k = 41$ nécessite 1 To de mémoire. Notre logiciel n'est donc pas capable de rechercher des grands k-mers ; il permet par contre de rechercher des séquences partageant plusieurs k-mers plus petits.

L'outil COMPAREADS est libre (sous licence CECILL) et disponible en ligne [1]. Notre outil a été publié dans BMC Bioinformatics [2] suite à la conférence RECOMB *Comparative Genomics 2012*. Le fonctionnement de la demi-intersection en programmé dans un logiciel codé en C, et la mise en place du pipeline complet consiste en un script Bash, qui utilise à trois reprises le logiciel de demi-intersection. COMPAREADS fonctionne sur tous les système UNIX, incluant GNU/Linux et Mac OS X.

3.3.2.2 *Choix des séquences*

Les scores fournis par COMAPREADS ne sont pas calculés sur l'intégralité des séquences traitées. En effet, certaines séquences ne sont pas utilisées par COMPAREADS. Plusieurs options permettent de filtrer les séquences à utiliser sur la base de leur longueur ou de leur composition. Filtrer les séquences sur leur taille permet de ne pas indexer des séquences sur lesquelles on ne peut trouver t k-mers distincts. Typiquement, avec les paramètres $t = 2$ et $k = 33$, les séquences de moins de 66 nucléotides ne seront pas utilisées, ni pour l'indexation, ni pour la recherche. COMPAREADS peut aussi filtrer les séquences sur la base de leur composition. En effet, certaines séquences sont composées de répétitions en tandem, c'est à dire deux ou plus copies adjacentes d'un motif donné de nucléotides (par exemple AAAAAA ou encore TATATA) [139]. Les NGS produisant généralement des séquences de quelques centaines de nucléotides, certaines séquences peuvent être composées exclusivement de répétitions. Dès lors, toutes les séquences contenant une partie de ces répétitions peuvent être considérées comme similaires par COMPAREADS. Pour pallier cet effet, le logiciel applique l'entropie de shannon [140] pour repérer les portions d'ADN les moins informatives et ignorer ces séquences. L'entropie de shannon a été implémentée pour quantifier l'entropie de séquences composées de A, C, G,

1. http ://colibread.inria.fr/software/compareads/
2. http ://www.biomedcentral.com/content/pdf/1471-2105-13-S19-S10.pdf

T ou N, N désignant un nucléotide non déterminé. Ainsi, une séquence composée d'une seule lettre répétée (par exemple AAAAAA) aura un indice de shannon de 0. Une séquence avec une composition équirépartie en A, C, G, T ou N aura un indice de shannon maximal égal à $\log_2(5) = 2{,}32$. Le logiciel peut ainsi filtrer les séquences afin de n'utiliser que celle supérieure à un seuil u, tel que : $0 \leq u \leq 2{,}32$.

CONCLUSION SUR L'OUTIL COMPAREADS Le logiciel COMPAREADS est capable de comparer deux jeux de données métagénomiques en utilisant une faible quantité de mémoire. Cette comparaison se base sur le nombre de séquences que chaque jeu partage avec l'autre jeu. Les séquences identifiées comme similaires entre les deux jeux ont une faible chance d'être des faux positifs, mais il n'y a pas de faux négatifs. Ces séquences similaires sont retournées par le logiciel. Un score de similarité global entre les deux jeux est fourni par COMPAREADS. Ce score est basé sur le nombre de séquences que les jeux de données partagent entre eux.

COMPAREADS est un des premiers logiciels dédié à la comparaison de grands jeux de données métagénomiques. De plus, à l'inverse de logiciels comme CRASS [105], il ne nécessite aucun logiciel tiers en amont. Dans le chapitre suivant, on analyse les performances de la structure de données et de l'outil face à divers logiciels permettant d'obtenir une mesure de similarité proche de celle de COMPAREADS. Puis, on teste cette méthode sur des jeux de données réels.

3.4 CONCLUSION

Le BDS est une nouvelle structure de données qui autorise l'indexation rapide des k-mers d'une grande quantité de données, pour un coût mémoire faible. De plus, il est très rapide de rechercher dans cet index si une donnée y est présente ou non. Le BDS n'admet aucun faux négatif, mais génère un faible pourcentage de faux positifs.

Dans notre méthodologie de métagénomique comparative *de novo*, on souhaite comparer deux échantillons sur la base du nombre de séquences qu'ils partagent. La première étape consiste à indexer tous les k-mers d'un des deux jeux de données. La seconde étape consiste à parcourir cet index pour identifier les séquences requêtes qui ont des k-mers présents dans cet index. Le BDS est particulièrement bien adapté à cette tâche.

Nous avons implémenté notre méthodologie dans un logiciel nommé COMPAREADS. Ce logiciel identifie et extrait toutes les séquences proches entre deux jeux de données métagénomiques mais génère un faible taux de faux positifs. À partir de ces séquences, il dérive un score de similarité. On verra, section 4.2, que ce score est biologiquement pertinent et que COMPAREADS est plus rapide que les autres méthodes permettant d'effectuer ce genre d'opérations. COMPAREADS a été utilisé sur plusieurs projets métagénomiques de grande envergure et les résultats sont présentés sous-section 4.3.

4

RÉSULTATS

Dans ce chapitre, on présente les résultats et performances de l'implémentation de notre méthodologie. Tout d'abord, la structure de données BDS est comparée à d'autre structures d'indexation. Ensuite, le logiciel COMPAREADS est confronté à différents outils. Le but est de tester les performances en temps, mémoire et résultats de notre méthode face à différentes approches existantes. Finalement, COMPAREADS est appliqué sur des jeux de données provenant de réels projets métagénomiques.

Dans la première section (voir 4.1), on analyse les performances du BDS, en temps, mémoire et résultats, comparé à différentes structure de données. La deuxième section (voir 4.2) étudie les performances de COMPAREADS, tant en terme de qualité de résultats que de ressources utilisées. La troisième section (voir 4.3) présente les résultats de COMPAREADS sur différentes expériences métagénomiques. La première étude (voir sous-section 4.3.1) concerne des petits métagénomes non publiés. La deuxième étude (voir sous-section 4.3.2) provient d'une publication récente sur le métagénome intestinal de l'escargot [98]. La troisième étude (voir sous-section 4.3.3) reproduit l'expérience du projet Metasoil [120], présenté sous-section 2.2.2. La quatrième étude (voir sous-section 4.3.4) s'intéresse au projet GOS [94] présenté sous-section 2.2.3. Finalement, la cinquième et dernière étude (voir sous-section 4.3.5) présente des résultats non publiés obtenus à l'aide de COMPAREADS sur plusieurs échantillons du projet Tara oceans.

4.1 PERFORMANCES DU BDS

Dans cette section, une analyse comparative du BDS avec d'autres structures de données est présentée. Dans ce qui suit, on montre que les structures non-probabilistes classiques ont de moins bonnes performances en temps ou en mémoire. Ainsi, il va être expliqué pourquoi le BDS est la structure la plus adaptée pour le problème de métagénomique comparative *de novo* dans lequel on cherche à identifier quelles séquences sont partagées entre deux jeux de données.

4.1.1 Comparaison avec d'autres structures de données

Pour rappel, le BDS fonctionne avec quatre fonctions de hachage particulières et permet d'indexer jusqu'à 1 milliard de 33-mers distincts en utilisant 4 Go de mémoire, pour un taux de faux positifs de 0,1% lors de la phase de requête.

4.1.1.1 *Tableau des suffixes*

Indexer une séquence de n caractères en utilisant l'implémentation originale des tableaux des suffixes, nécessite au minimum 40 bits/caractères, soit $5n$ octets [125, 129]. En indexant une séquence d'un milliard de nucléotides, l'empreinte mémoire d'un arbre des suffixes serait de 5×10^9 octets, soit 4,66 Go.

Dans notre utilisation du BDS, on indexe 1 milliard de 33-mers. Sur une séquence de 100 pb, il y a 68 33-mers ; le BDS peut donc contenir tous les 33-mers de $\frac{10^9}{68}$ séquences de 100 pb. Dans le tableau des suffixes, on utilise un caractère unique pour séparer chaque séquence à indexer. On doit donc indexer une séquence totale de $\frac{10^9}{68} \times 101$ caractères. À raison de 5 octets par caractère, l'empreinte mémoire d'un tableau des suffixes est alors de $\frac{10^9}{68} \times 101 \times 5$ octets, soit environ 6,92 Go.

La consommation mémoire d'un tableau des suffixes est donc supérieur aux 4 Go utilisés par le BDS. Par contre, cette structure présente l'avantage d'avoir une empreinte mémoire qui ne dépend pas de la taille des k-mers et le résultat de la recherche est exact, il n'y a pas de faux positifs.

4.1.1.2 *Table de hachage*

Une table de hachage peut être utilisée pour stocker de manière exacte un ensemble de k-mers. Une telle structure stocke les k-mers explicitement. En utilisant le codage classique de l'ADN en binaire, soit 2 bits par nucléotide, il faut $n \cdot \left\lceil \frac{2k}{8} \right\rceil \cdot 8$ bits, soit 8,38 Go pour stocker 1 milliard de 33-mers ($\left\lceil \frac{2k}{8} \right\rceil$ étant l'arrondi à l'octet supérieur du stockage d'un k-mer). Pour être tout à fait exact, un k-mer ne peut être représenté que sur une puissance de 2 bits. On a alors besoin de $n \cdot 2^{\lceil \log_2(2k) \rceil}$ bits, soit 14,90 Go pour stocker 1 milliard de 33-mers ($2^{\lceil \log_2(2k) \rceil}$ étant l'arrondi à la puissance de 2 bits supérieur du stockage d'un k-mer). Ce calcul ne prend pas en compte les listes chaînées pour pallier les collisions. Le BDS est donc près de quatre fois plus léger en mémoire, mais la table de hachage ne génère pas de faux positifs.

4.1.2 Comparaison avec d'autres fonctions de hachage et un filtre de Bloom

Les fonctions de hachage utilisées dans le BDS ont été créées dans un but de rapidité. Elles sont ici comparées en terme de temps d'exécution et de faux positifs avec d'autres fonctions couramment utilisées : les fonctions de Jenkins. Ces fonctions, particulièrement hashlittle2 de lookup3 [1], sont des fonctions de hachage souvent utilisées pour leurs bonnes performances en terme de vitesse.

4.1.2.1 *Comparaison du temps d'exécution*

1 million de séquences de 100 paires de base (pb) ont été simulées, où chaque nucléotide a été tiré indépendamment selon une loi uniforme. Utilisant les quatre fonctions de hachage du BDS, quatre codes de hachage ont été calculés pour chaque 33-mer. De même, quatre codes de hachage par 33-mers ont été calculés avec la fonction hashlittle2, en utilisant quatre valeurs d'initialisation différentes. La moyenne du temps de calcul total pour tous les 33-mers a été calculée sur trois exécutions. Calculer les codes de hachage avec la fonction hashlittle2 a pris 13,1s, soit 5,2 MHashs/s. Les fonctions du BDS ont mis 1,4s, soit 49,8 MHashs/s pour remplir la même tâche. Ceci permet de confirmer que les fonctions de hachage du BDS sont plus rapides d'un ordre de grandeur comparées aux fonctions de hachage plus classiques.

4.1.2.2 *Comparaison du taux de faux positifs*

Comme montré figure 25, le taux de faux positifs des fonctions de Jenkins suit parfaitement le taux de faux positifs des fonctions équilibrées. Les fonctions de Jenkins suivent l'équation 12,

1. Fonction disponible à l'adresse suivante : http ://burtleburtle.net/bob/c/lookup3.c

$P_{FP}(f_1 \cap f_2 \cap f_3, k, n) \lesssim (1 - (1 - \frac{1}{2^k})^n)^h$, où h est le nombre de fonctions utilisées. Comme seules trois fonctions équilibrées existent, la combinaison "BDS 4 fonctions" contient une fonction non-équilibrée et génère plus de faux positifs que les fonctions de Jenkins. Pour quatre fonctions et 1 milliard de 33-mers distincts indexés, quatre fonctions de Jenkins produisent en moyenne 0,01% de faux positifs quand les quatre fonctions du BDS en génèrent 0,114%. Bien que ce taux de faux positifs soit 10 fois plus élevé, il reste négligeable au regard de notre problématique, et le gain significatif au niveau temps d'exécution incite à utiliser les fonctions de hachage du BDS.

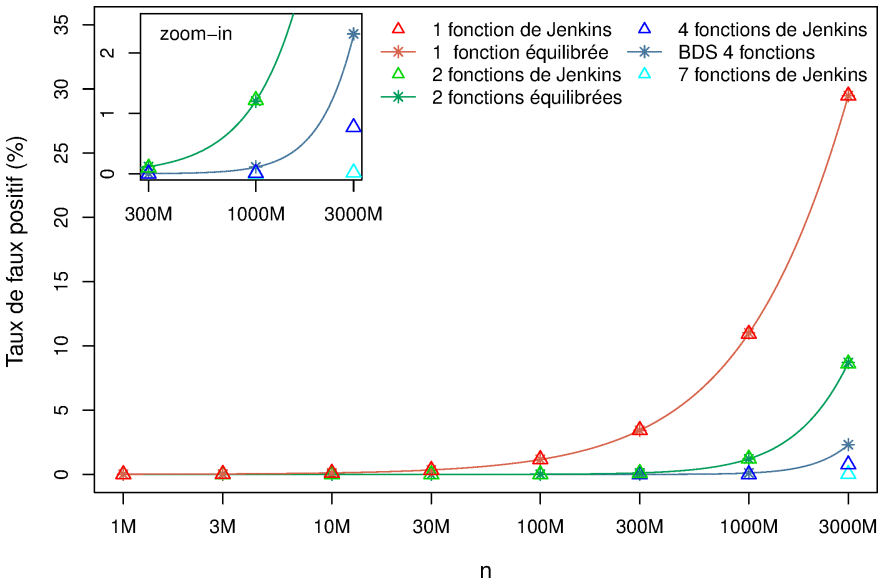


FIGURE 25: Taux de faux positifs pour les fonctions de hachage du BDS et des fonctions de Jenkins en fonction du nombre de k-mers indexés (échelle logarithmique). Les lignes correspondent aux prédictions théoriques pour les fonctions équilibrées du BDS et les étoiles et les triangles, aux observations empiriques obtenues par simulation respectivement pour les fonctions du BDS et de Jenkins. Les données ont été obtenues pour $k = 33$. La combinaison "4 fonctions" est composée des trois fonctions équilibrées et d'une fonction non-équilibrée.

4.1.2.3 *Comparaison avec un filtre de Bloom conventionnel*

Un filtre de Bloom classique nécessite une quantité de mémoire fixe pour indexer n k-mers. La mémoire nécessaire pour un filtre de Bloom contenant 1 milliard d'éléments avec un taux de faux positifs de 0,114% peut être évaluée à 1,64 Go en utilisant la formule section 2.3.5. Pour le même nombre d'éléments et le même taux de faux positifs, la consommation mémoire d'un filtre de Bloom est 2,5 fois plus faible que celle du BDS. Mais, un filtre de Bloom classique nécessiterait des fonctions de hachage classiques. Or, comme expliqué ci-dessus, les fonctions de hachage classiques sont plus lentes d'un ordre de grandeur.

CONCLUSION SUR LES PERFORMANCES DU BDS Un tableau des suffixes utilise une quantité de mémoire de même ordre de grandeur que le BDS, mais le temps de recherche dans cette structure est significativement moins bon. Le BDS est quatre fois plus économique en mémoire qu'une table de hachage. Le filtre de Bloom, à taux de faux positif égal, est 2,5 fois plus léger en mémoire mais 10 fois plus lent à construire et à interroger.

L'insertion d'éléments dans une table de hachage est souvent considérée comme étant en temps constant $O(1)$. De même, un filtre de Bloom mis en œuvre avec une seule fonction de Jenkins et le BDS n'utilisant qu'une seule fonction de hachage, sont supposés en temps constant $O(1)$ à la construction. En pratique, pour la table de hachage, le filtre de Bloom ou le BDS, ce

temps dépend de la taille des mots à hacher. Pour indexer tous les k-mers d'une séquence s, la table de hachage et le filtre de Bloom sont en $O(k \times |s|)$. Une des originalités du BDS est qu'il est particulièrement efficace pour calculer les codes de hachage de k-mers consécutifs. En effet, seul le caractère différent entre deux k-mers consécutifs est haché. Ainsi, le BDS indexe tous les k-mers d'une séquence s en $O(|s|)$, peut importe k.

Les deux besoins principaux en terme d'indexation dans le cadre de la méthode de métagénomique comparative *de novo* qu'on met en place, sont : I) de pouvoir indexer un très grand nombre de séquences, en utilisant peu de mémoire et le plus rapidement possible, et II) de rechercher dans cet index, le plus rapidement possible, un autre très grand nombre de séquences. Le BDS permet ceci avec un taux de faux positifs tout à fait raisonnable de 0,1% et est donc la structure la plus adaptée à la problématique, car il est soit plus léger, soit plus rapide que les autres structures sur l'opération fondamentale de notre méthode.

4.2 PERFORMANCES DE COMPAREADS SUR DES DONNÉES SIMULÉES

Dans cette section, on compare les résultats et les performances en temps et mémoire de six logiciels. Ces logiciels sont BLAT [103], BLAST [22], UBLAST [104], TriageTools [61], CRASS [105] et COMPAREADS. Ils ont tous été testés sur une même machine disposant de 512 Go de mémoire vive et de 80 processeurs. Les trois approches d'alignements, BLAT, BLAST et UBLAST, ont été configurées pour rechercher des alignements d'au moins 60 nucléotides (équivalent à deux 30-mers non chevauchants consécutifs) et un pourcentage d'identité de séquence supérieur à 80%. COMPAREADS a été testé avec deux jeux de paramètres différents. Le premier, nommé Compareads30, nécessite deux 30-mers en commun entre des séquences pour les déclarer similaires. Le second, Compareads33, utilise deux 33-mers. Finalement, TriageTools a été utilisé avec les paramètres par défaut ($k = 14$ et $H = 36$) et CRASS n'utilise pas de paramètre.

BLAST et UBLAST nécessitent de transformer un des jeux de données à traiter en une base de donnée. Dans la suite du document, le temps de création des bases de données est ajouté au temps de recherche. De plus, ces deux logiciels sont parallélisés. Par exemple, UBLAST utilise parfois plus de 70 des 80 processeurs disponibles. Néanmoins, le temps présenté dans les résultats pour tous les logiciels est celui qui se déroule réellement entre le début et la fin du programme (*Wallclock Time*), et non le temps total pris par tous les processeurs, ramené sur un seul processeur (*CPU Time*). Par exemple, si UBLAST met 1 minute à calculer les séquences similaires entre deux jeux de données en utilisant 50 processeurs, alors le temps utilisé dans ce document sera effectivement 1 minute et non 50 minutes sur 1 processeur. Concernant la mémoire, la création des bases de données de BLAST et UBLAST demande toujours une quantité plus faible de mémoire que le reste de l'exécution du logiciel ; cette quantité de mémoire n'est donc pas comptabilisée et seul le pic de mémoire des logiciels est présenté dans les comparaisons.

Le temps et la mémoire présentés pour CRASS tiennent compte du processus d'assemblage mené avec Newbler. Les quatre distances que CRASS délivre (voir sous-section 2.1.4) sont comprises entre 0 et 1, $d = 0$ indiquant des jeux identiques, et $d = 1$ indiquant des jeux totalement différents. Les quatre notions de similarités pour CRASS dans le reste du document seront définies comme $\forall x \in \{1,2,3,4\}, s_x = (1 - d_x) \times 100$ afin d'être comparables à la notion de similarité de COMPAREADS. Les auteurs de CRASS précisent qu'ils espèrent que chaque distance se révélera utile pour un cas d'utilisation donné. Pour tous les autres logiciels, le score de similarité global est calculé de la même manière que dans COMPAREADS (voir 3.2.1.2).

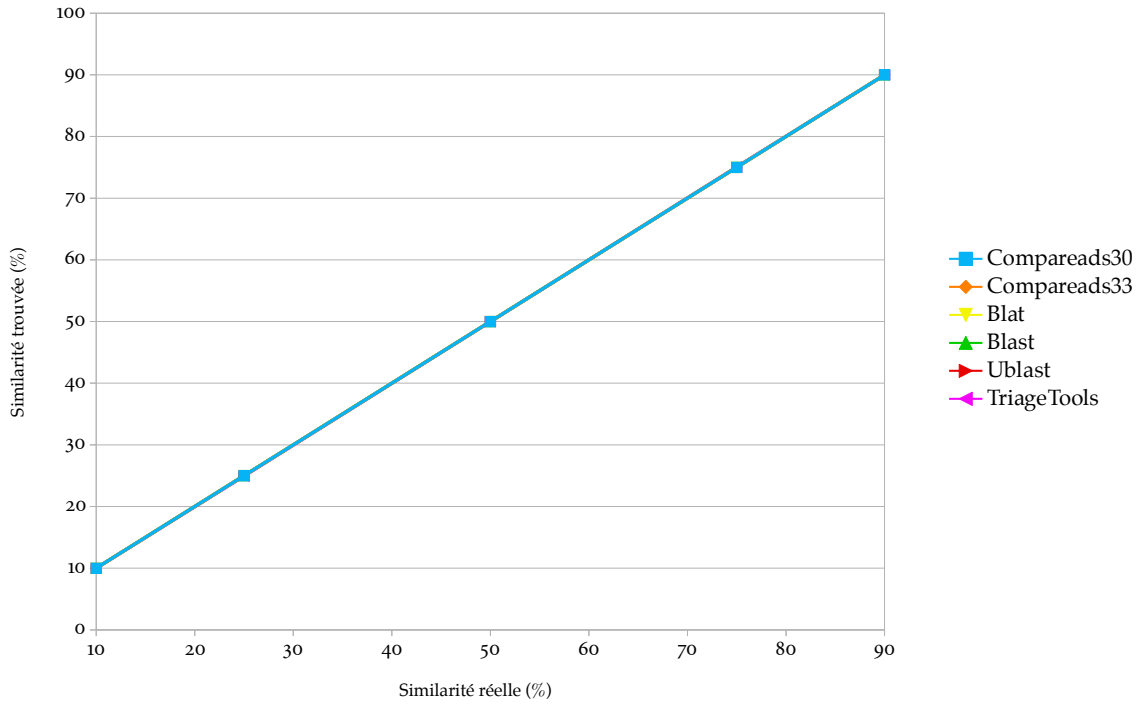


FIGURE 26: Représentation de la similarité trouvée par les différents logiciels en fonction de la vraie similarité entre les deux échantillons. Les jeux de données sont composés de 100 000 séquences de 100 pb. Les courbes se superposent.

Définitions

A. SÉQUENCES IDENTIQUES : Deux séquences s_1 et s_2 sont *identiques* si et seulement si $\forall i \in [0, \max(|s_1|, |s_2|) - 1], s_1[i] = s_2[i]$. On a alors $s_1 = s_2$.

B. SÉQUENCE UNIQUE : Une séquence s est *unique* dans un ensemble E de séquences si et seulement si $\forall e \in E \setminus s, e \neq s$.

4.2.1 Performances sur des séquences simulées

Dans cette partie, on compare des jeux de données totalement simulés afin d'en contrôler les différents paramètres. Pour chaque comparaison, nous avons créé deux jeux de données où chaque séquence est unique, mis à part un pourcentage donné. Ce pourcentage est constitué de paires de séquences identiques entre les deux jeux et est appelé **similarité réelle**. Ainsi, chaque séquence est unique par jeu de données, et le taux de séquences similaires entre les deux jeux est maîtrisé. Le logiciel CRASS n'a pas pu être lancé. Il nécessite une phase d'assemblage qui échoue systématiquement avec des séquences ainsi simulées.

4.2.1.1 *Évolution du nombre de séquences similaires*

Le premier test, dont les résultats sont présentés figure 26, vérifie que tous les outils retrouvent chacun les séquences identiques entre deux jeux de données de 100 000 séquences de 100 paires de bases chacune, peu importe le pourcentage de similarité réelle. Toutes les courbes se superposent, tous les logiciels retrouvent bien la similarité réelle.

Ces logiciels utilisent chacun une quantité constante de mémoire, peu importe la similarité entre les jeux de données (voir figure 27). Les logiciels BLAST et TRIAGETOOLS utilisent le plus de mémoire, respectivement 6,1 GO et 7,0 GO. COMPAREADS utilise 715 MO en cherchant des 30-

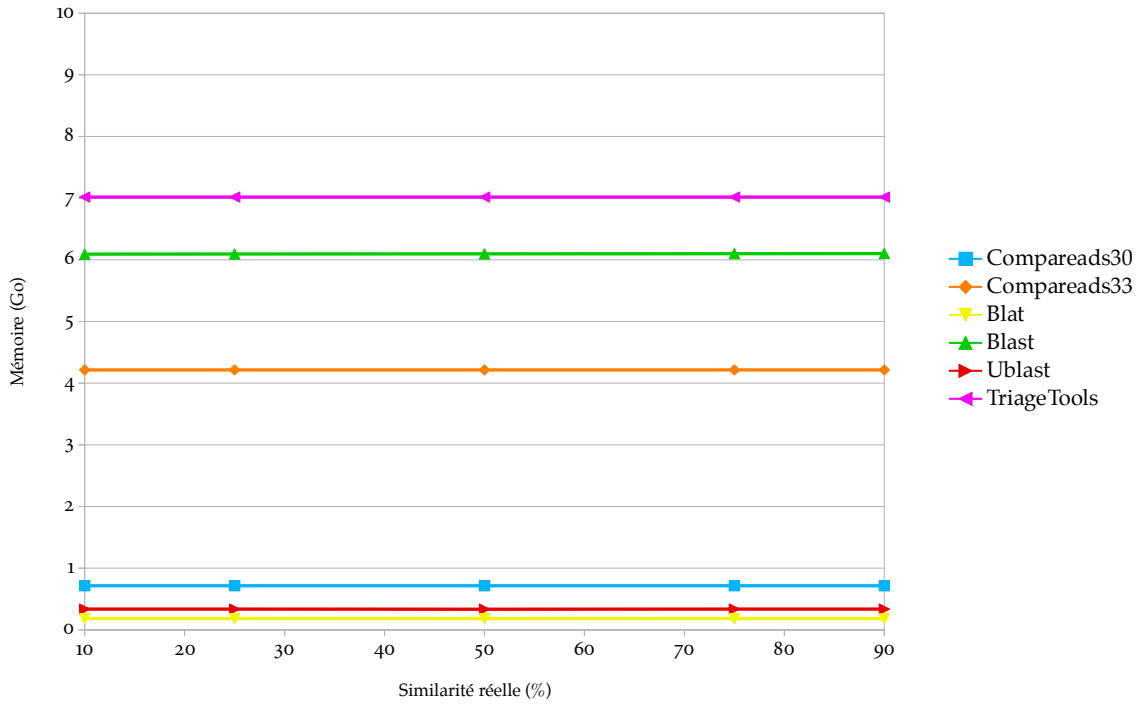


FIGURE 27: Représentation de l'utilisation mémoire des différents logiciels en fonction de la similarité entre les deux échantillons. Les jeux de données sont composés de 100 000 séquences de 100 pb.

mers et 4,2 Go pour des 33-mers. UBLAST et BLAT sont les plus légers en mémoire et nécessitent respectivement 337 Mo 182 Mo.

Le temps d'exécution des logiciels a tendance à augmenter en fonction du nombre de séquences identiques (voir figure 28). Le logiciel TRIAGETOOLS semble prendre moins de temps quand la similarité est à 90%. Les résultats pour BLAST ne sont pas montrés pour la lisibilité : il lui faut plus de 13 minutes par expérience.

Ce premier test permet de valider que COMPAREADS est capable de retrouver des séquences identiques entre deux échantillons. De plus, son impact mémoire et son temps d'exécution sont comparables aux autres solutions. Le logiciel BLAT est le plus rapide et le plus économe en mémoire sur ces jeux de données, ce qui n'est pas le cas sur plusieurs autres expériences montrées dans la suite du manuscrit. UBLAST et TRIAGETOOLS sont les deux solutions les plus proches de COMPAREADS en temps et en mémoire. Finalement, BLAST fonctionne correctement mais nécessite à la fois beaucoup de temps et de mémoire.

4.2.1.2 *Évolution de la taille des séquences*

Le second test consiste à faire évoluer la taille des séquences. Chaque jeux de données comporte 100 000 séquences. Il y a 25% de séquences identiques entre tous les jeux pris deux à deux. La taille des séquences a été fixée à cinq valeurs représentatives des différentes technologies de séquençage couramment utilisées : des séquences de 50 pb et 100 pb pour SOLID ou ILLUMINA, 200 pb pour ION TORRENT, 400 pb pour ROCHE 454 et 1000 pb pour SANGER. La figure 29 montre les résultats de la similarité trouvée. COMPAREADS ne peut pas être lancé en recherchant 2 30-mers ou 2 33-mers sur des séquences de 50 pb. On remarque que tous les logiciels sauf TRIAGETOOLS se comportent comme attendu. Pour ce logiciel, on voit, en utilisant l'équation 1 avec $k = 14$, $H = 36$, $T = 1000 \times 100\,000$ et $R = 1000$, qu'indexer 100 000 séquences de 1000 pb avec les paramètres par défaut conduit à 100% de faux positifs.

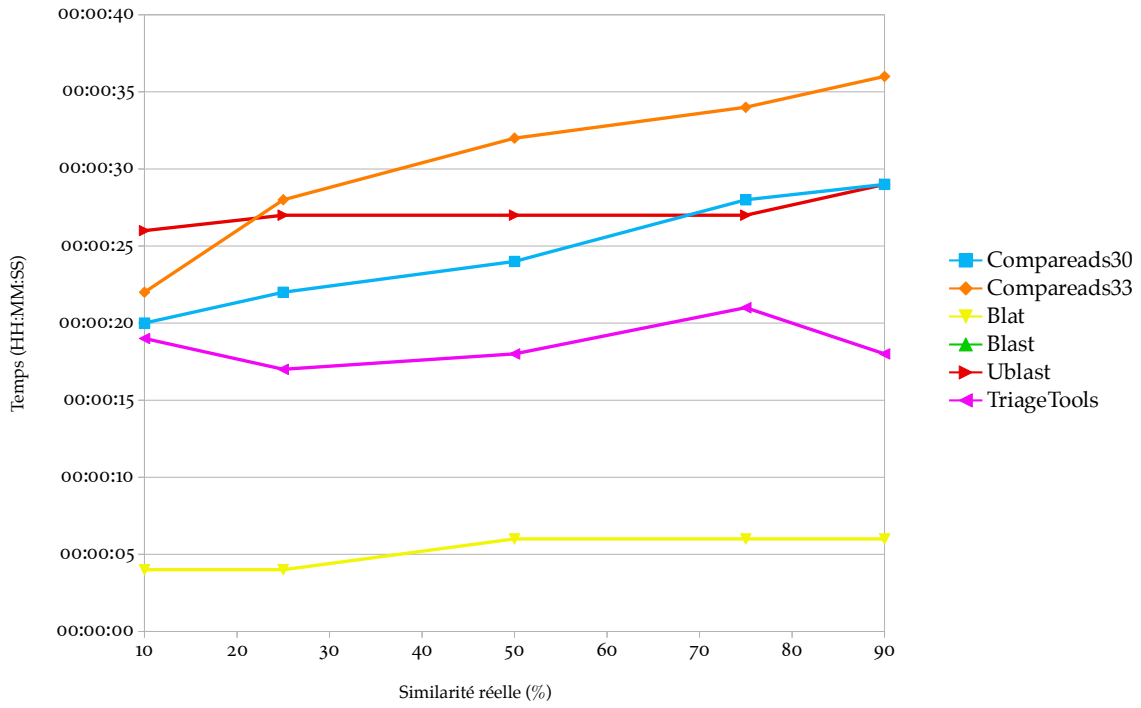


FIGURE 28: Représentation du temps de fonctionnement des différents logiciels en fonction de la similarité entre les deux échantillons. Les jeux de données sont composés de 100 000 séquences de 100 pb.

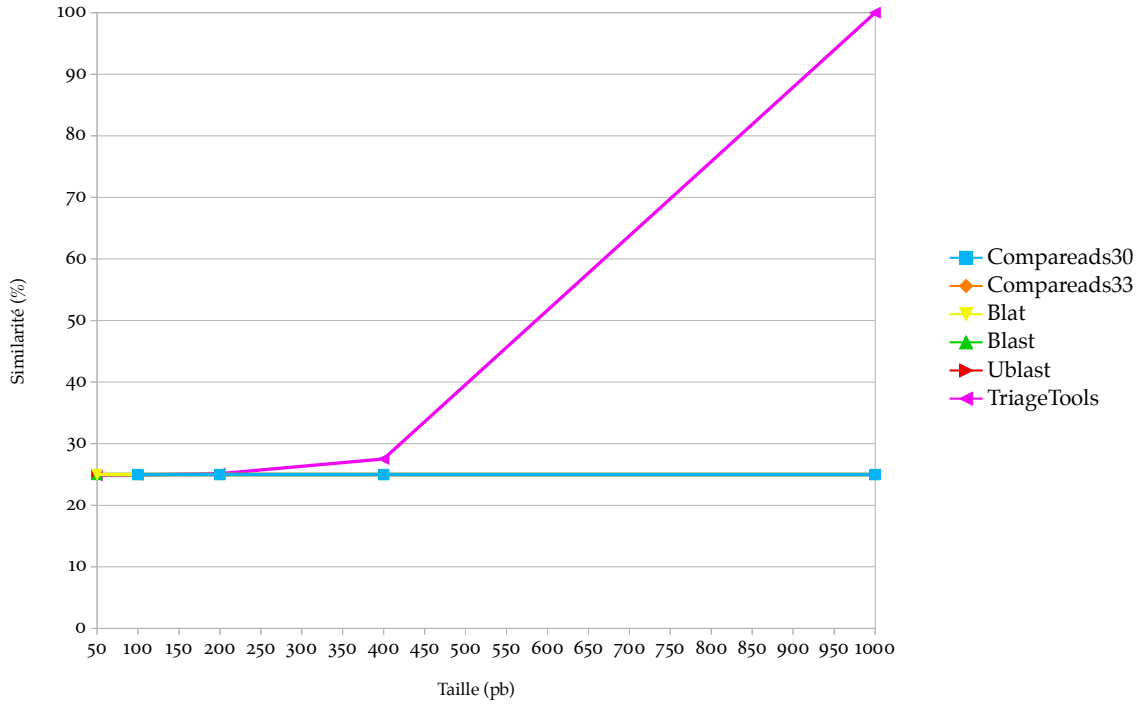


FIGURE 29: Représentation de la similarité trouvée par les différents logiciels en fonction de la taille des séquences. Les jeux de données sont composés de 100 000 séquences dont 25% sont identiques. Toutes les courbes se superposent, sauf celle de TRIAGETOOLS.

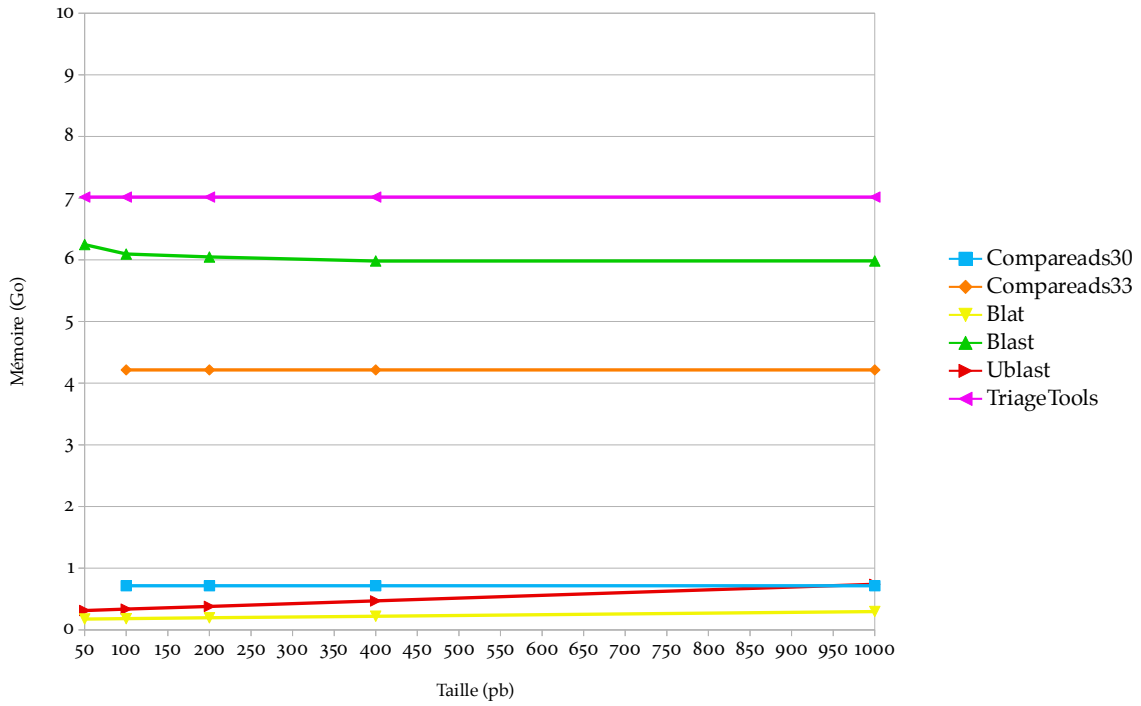


FIGURE 30: Représentation de l'utilisation mémoire des différents logiciels en fonction de la taille des séquences. Les jeux de données sont composés de 100 000 séquences dont 25% sont identiques.

Mis à part BLAT et UBLAST, les logiciels utilisent chacun une quantité globalement constante de mémoire, peu importe la taille des séquences (voir figure 30). Les logiciels BLAT et UBLAST voient leur utilisation mémoire augmenter, passant de 176 MO à 297 MO pour BLAT et de 314 MO à 739 MO pour UBLAST. Ces deux logiciels restent tout de même les plus économes, en terme de mémoire. TriageTools et COMPAREADS sont constants en mémoire : leur utilisation mémoire est donc la même que lors du premier test.

Le temps d'exécution des logiciels augmente avec la taille des séquences (voir figure 31). Les résultats pour BLAST sont tronqués pour la lisibilité : il lui faut 3h36m34s pour traiter les séquences de 1000 pb. BLAT, TriageTools et COMPAREADS ont une croissance linéaire. UBLAST croît bien plus vite avec la taille des séquences et nécessite 34m32s pour traiter les séquences de 1000 pb. Compareads30 et Compareads33 sont légèrement plus rapides que les autres approches pour traiter les plus grandes séquences.

Ce test valide le fonctionnement de COMPAREADS sur des séquences d'une taille de 100 à 1000 pb. Son impact mémoire est stable, comme plusieurs autres logiciels. Son temps d'exécution est parmi les meilleurs, avec BLAT et TriageTools.

4.2.1.3 *Évolution du nombre de séquences*

Le dernier test consiste à tester le passage à l'échelle en terme de nombre de séquences. Chaque jeu de données comporte plusieurs milliers de séquences de 100 pb. Il y a 25% de séquences identiques entre tous les jeux pris deux à deux. La figure 32 montre les résultats de la similarité trouvée. Comme pour la taille des séquences, TriageTools se dégrade rapidement quand la quantité de données est importante. L'équation 1 indique en effet 100% de faux positifs à partir de 10 millions de séquences de 100 pb. COMPAREADS se comporte bien, mais la version utilisant les 30-mers trouve 25,91% de similarité globale entre les deux jeux de données de 100 millions de séquences. La similarité réelle est de 25,00% et le taux de faux

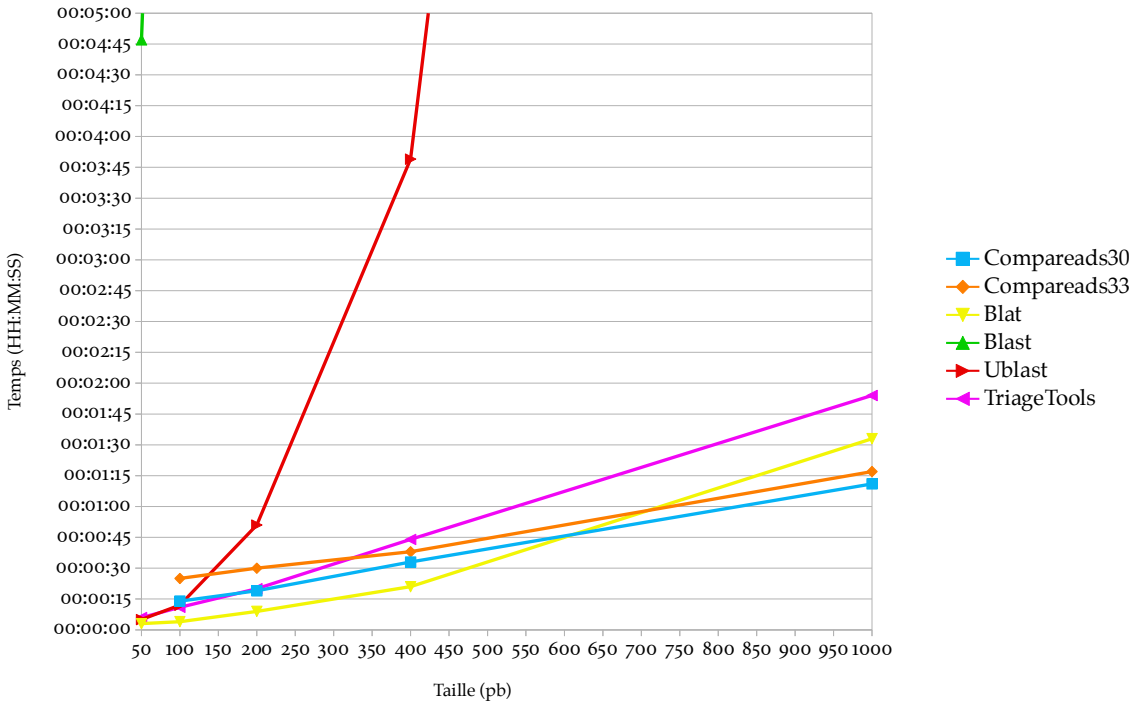


FIGURE 31: Représentation du temps de fonctionnement des différents logiciels en fonction de la taille des séquences. Les jeux de données sont composés de 100 000 séquences dont 25% sont identiques.

positif est alors bien supérieur aux 0,1% attendus du BDS.

Pour comprendre cet effet, il faut se souvenir que ce 0,1% correspond au taux théorique de faux positifs du BDS. La phase d'indexation est interrompue quand le volume de k-mers indexés dans le BDS dépasse un certain seuil. La phase de requête traite alors toutes les séquences à comparer, et chaque séquence a une probabilité de 0,1% ($p = 0{,}001$) d'être un faux positif du BDS. Autrement dit, une séquence a une probabilité $p = 0{,}999$ de ne *pas* être un faux positif du BDS. Quand la phase de requête est terminée, un nouveau BDS est créé par la seconde phase d'indexation. Lors de la seconde phase de requête, une séquence a de nouveau une probabilité $p = 0{,}999$ de ne pas être un faux positif du second BDS. Les deux BDS sont indépendants ; une séquence a donc une probabilité $p = 0{,}999^i$, avec $i = 2$, de ne pas être un faux positif ni du premier, ni du second BDS. Alors, une séquence a une probabilité $p = 1 - 0{,}999^i$ d'être un faux positif dans au moins un BDS parmi i.
Pour indexer tous les 30-mers de 100 millions de séquences de 100 pb, on doit traiter 7,1 milliards de k-mers. En utilisant l'équation 13, on remarque que le BDS peut contenir 113 millions de k-mers maximum. On a donc besoin de 63 BDS pour indexer la totalité des 30-mers. Chaque séquence a alors une probabilité $p = 1 - 0{,}999^{63}$, soit 6,1%, d'être un faux positif dans au moins un des BDS lors du calcul de $(A \overset{\rightrightarrows}{\cap} B)^*$. Les différentes stratégies mises en place dans COMPAREADS (voir sous-section 3.2.2.3) réduisent le taux de faux positifs final des résultats $(A \overset{\rightrightarrows}{\cap} B)$ et $(B \overset{\rightrightarrows}{\cap} A)$: sur le test précédent, ce taux est à 0,91%.

En utilisant des 33-mers (Compareads33), le BDS peut contenir 1 milliard de k-mers. On a alors besoin de 7 BDS pour indexer les 6,8 milliards 33-mers de 100 millions de séquences de 100 pb. Chaque séquence à comparer a alors une probabilité $p = 1 - 0{,}999^7$, soit 0,7%, d'être un faux positif dans un des BDS. On voit alors, figure 32, que le taux de faux positifs final est de 0,04%, bien plus faible que pour Compareads30.

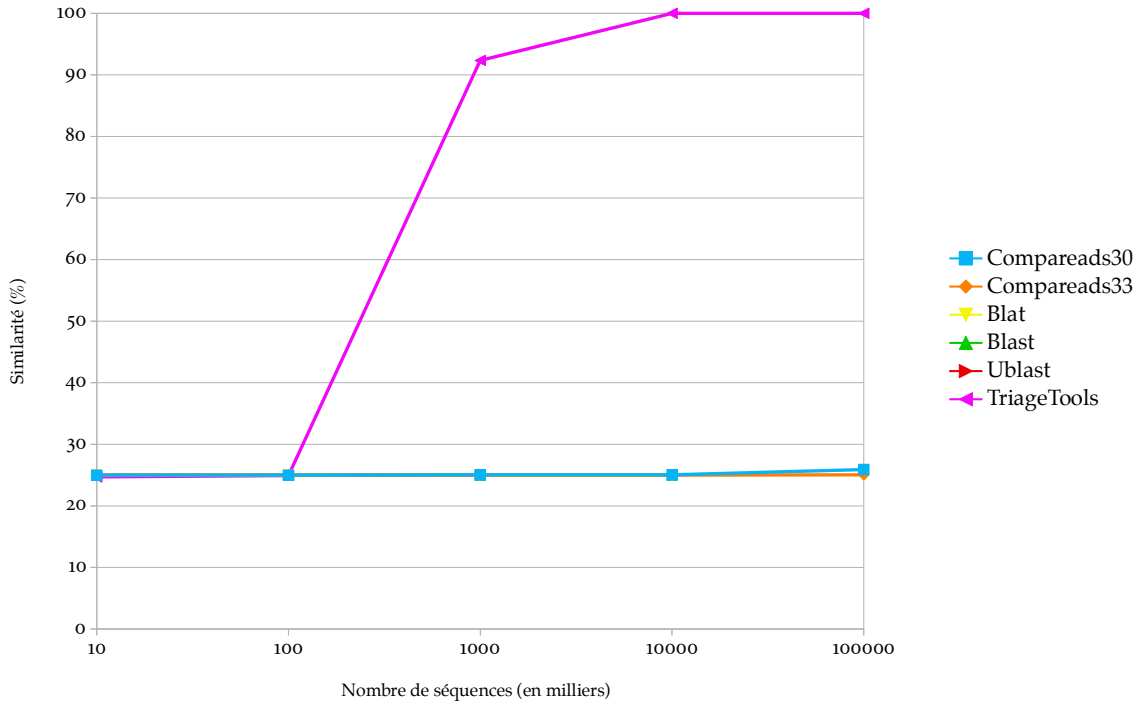


FIGURE 32: Représentation de la similarité trouvée par les différents logiciels en fonction du nombre de séquences (échelle logarithmique). Les jeux de données ont 25% de séquence identiquess de 100 pb. Seuls TRIAGETOOLS et COMPAREADS, sont à même de finir les calculs.

Finalement, il est important de noter que seul TRIAGETOOLS et COMPAREADS arrivent à traiter les jeux de 100 millions de séquences. BLAT produit une erreur après 29,9 GO de RAM utilisée et UBLAST après 11,1 GO. Les données ne sont donc pas visibles : il est impossible d'évaluer la quantité de mémoire que ces deux logiciels auraient utilisé pour finir les calculs. Le logiciel BLAST a été stoppé après plus d'une semaine de calcul.

Comme montré figure 33, TRIAGETOOLS et COMPAREADS sont constants en mémoire. BLAST, UBLAST et BLAT augmentent avec la quantité de données à traiter et finissent par ne plus fonctionner.

Le temps d'exécution des logiciels augmente avec la taille des séquences (voir figure 34). Le temps d'exécution de l'approche Compareads30 se dégrade après 10 millions de séquences par jeux de données. Comme expliqué précédemment (voir sous-section 3.2.3.4), le partitionnement de l'index conduit à tester toutes les séquences non-similaires avec chaque partition de l'index. Pour $k = 30$, on a besoin de 63 BDS pour indexer 100 millions de séquences de 100 pb lors du calcul de $(A \overset{\rightsquigarrow}{\cap} B)^*$; 75% des séquences sont uniques (donc non-similaires), et alors testées 63 fois lors de la première passe, ce qui ralentit l'exécution. Pour Compareads33, il suffit de 7 BDS pour indexer les 100 millions de séquences : le calcul est donc beaucoup plus rapide. On remarque que TriageTools est près de 3 fois plus rapide que COMPAREADS, mais ses résultats ne sont pas concluants.

Ce troisième test valide le fonctionnement de COMPAREADS sur de grands jeux de données. Son impact mémoire est stable. Le logiciel TRIAGETOOLS est plus rapide mais consomme deux fois plus de mémoire, pour des résultats inexploitables.

CONCLUSION Ce premier ensemble de tests confirme que COMPAREADS est capable de retrouver toutes les séquences identiques entre deux jeux, et ce, peu importe la taille des séquences ou leur quantité. Seuls COMPAREADS et TRIAGETOOLS sont capables de gérer de gros

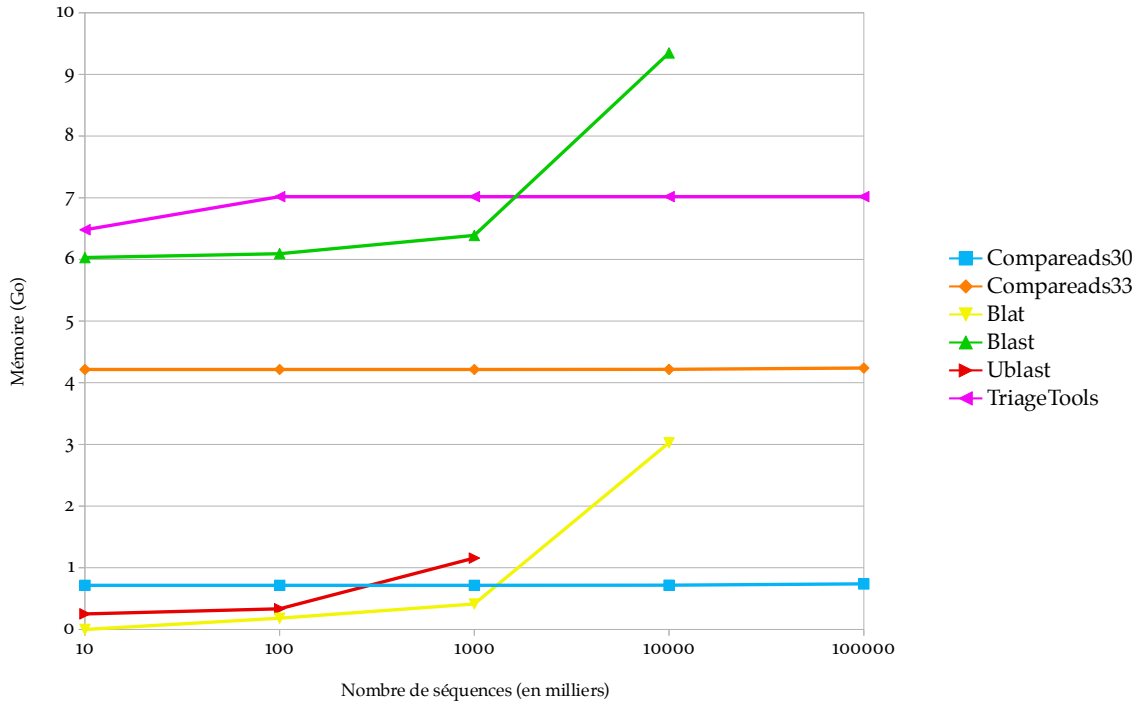


FIGURE 33: Représentation de l'utilisation mémoire des différents logiciels en fonction du nombre de séquences (échelle logarithmique). Les jeux de données ont 25% de séquences identiques de 100 pb.

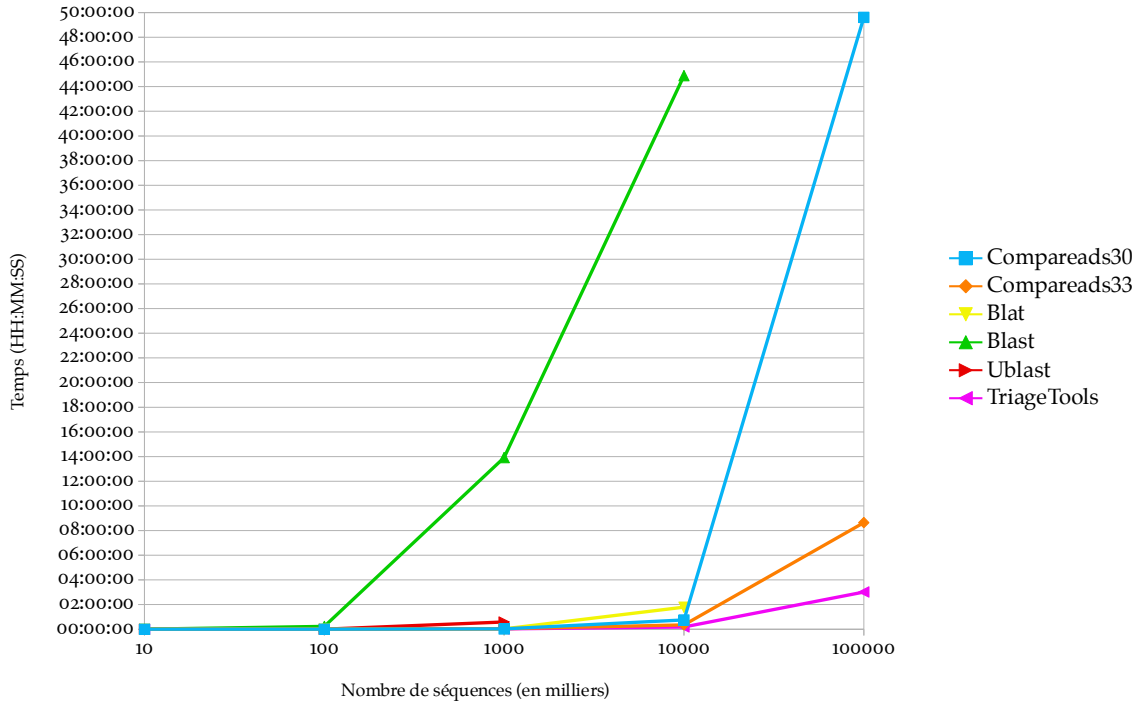


FIGURE 34: Représentation du temps de fonctionnement des différents logiciels en fonction du nombre de séquences (échelle logarithmique). Les jeux de données ont 25% de séquences identiques de 100 pb.

jeux de données de la taille d'un métagénome, *i.e.* 100 millions de séquences de 100 bp. En terme de qualité de résultats, COMPAREADS surestime un peu la quantité de séquences à retrouver, ceci à cause des faux positifs. En revanche, TRIAGETOOLS n'est pas capable, sur les jeux de données simulés utilisés ici, d'estimer correctement la similarité réelle.

4.2.2 Performances sur des métagénomes simulées

Dans cette sous-section, on compare des jeux de données partiellement contrôlés et plus réalistes d'un point de vue séquence. 19 génomes bactériens ont d'abord été sélectionnés, venant d'ordres différents. Les génomes contenant des nucléotides indéterminés ('N' dans leur ADN) et moins de 2 millions de paires de bases ont été éliminés. Chaque génome a été séquencé *in silico* à 10x de couverture, de manière uniforme et sans erreur. Les jeux de données ainsi obtenus ont été analysés les uns contre les autres à l'aide de BLAT et de COMPAREADS. Comme expliqué sous-section 4.2.1, BLAT produit de très bons résultats sur des petits jeux de données et est plus rapide que BLAST ou UBLAST, aux résultats semblables. Seuls les génomes ayant moins de 0,02% de séquences similaires à un des autres génomes utilisés ont été conservés : 12 génomes ont passé ce filtre. Seul le premier million de paires de bases de ces 12 génomes a été conservé pour la suite des expérimentations. On dispose ainsi de 12 séquences complètes (sans 'N') de génomes de même taille et très éloignées, à la fois d'un point de vue séquence et d'un point de vue biologique. Le logiciel CRASS est utilisable sur ces données car la phase d'assemblage peut être réalisée.

4.2.2.1 *Évolution de la taille des séquences*

Une première expérience consiste à vérifier le comportement des logiciels sur la seule base de la taille des séquences des jeux de données. Le précédent test sur la taille des séquences utilisait un nombre constant de séquences ; ces séquences étaient donc de plus en plus grandes. Ici, le nombre de paires de bases est constant, donc plus les séquences sont grandes, moins elles sont nombreuses. Ce test s'apparente à séquencer un même métagénome avec différentes techniques de séquençage, à couverture constante.
Pour cela, deux métagénomes sont créés, contenant chacun 3 génomes d'un million de paires de bases chacun. Les deux métagénomes ont deux génomes en commun. Par exemple, le premier métagénome contient les génomes A, B et C et le second métagénomes, les génomes A, B et D. On dit alors que ces deux métagénomes **partagent** deux génomes. Les métagénomes sont séquencés *in silico* avec une couverture de 5x pour obtenir des séquences de 100 pb, 200 pb, 400 pb et 1000 pb. On a donc des jeux de données de chacun 150 000 séquences de 100 pb, des jeux de 75 000 séquences de 200 pb, des jeux de 37 500 séquences de 400 pb et des jeux de 15 000 séquences de 1 000 pb. La figure 35 montre les résultats de la similarité trouvée par les logiciels entre les jeux de données de même taille de séquence (100 pb contre 100 pb, 200 pb contre 200 pb, etc). On remarque que tous les logiciels trouvent une similarité proche de la similarité réelle (66,66%). Pour CRASS, la distance "Reads" est la plus proche de l'attendu, les trois autres distances donnant des scores de similarité inférieurs à 50%.

Comme pour l'expérience précédente sur la taille des séquences, les logiciels utilisent une quantité globalement constante de mémoire (voir figure 36). Seul BLAST montre une diminution de sa consommation mémoire, passant de 6,8 Go à 6,0 Go, contrairement aux précédentes observations sur la taille des séquences. Ce n'est donc pas tant la taille des séquences que le nombre de séquences à traiter qui influe sur l'utilisation mémoire de BLAST. UBLAST, BLAT, CRASS et Compareads30 sont les solutions utilisant le moins de mémoire.

Comme on peut le voir figure 37, le temps d'exécution des logiciels n'évolue pas de la même manière pour tous. UBLAST, TRIAGETOOLS et COMPAREADS sont globalement constant en temps

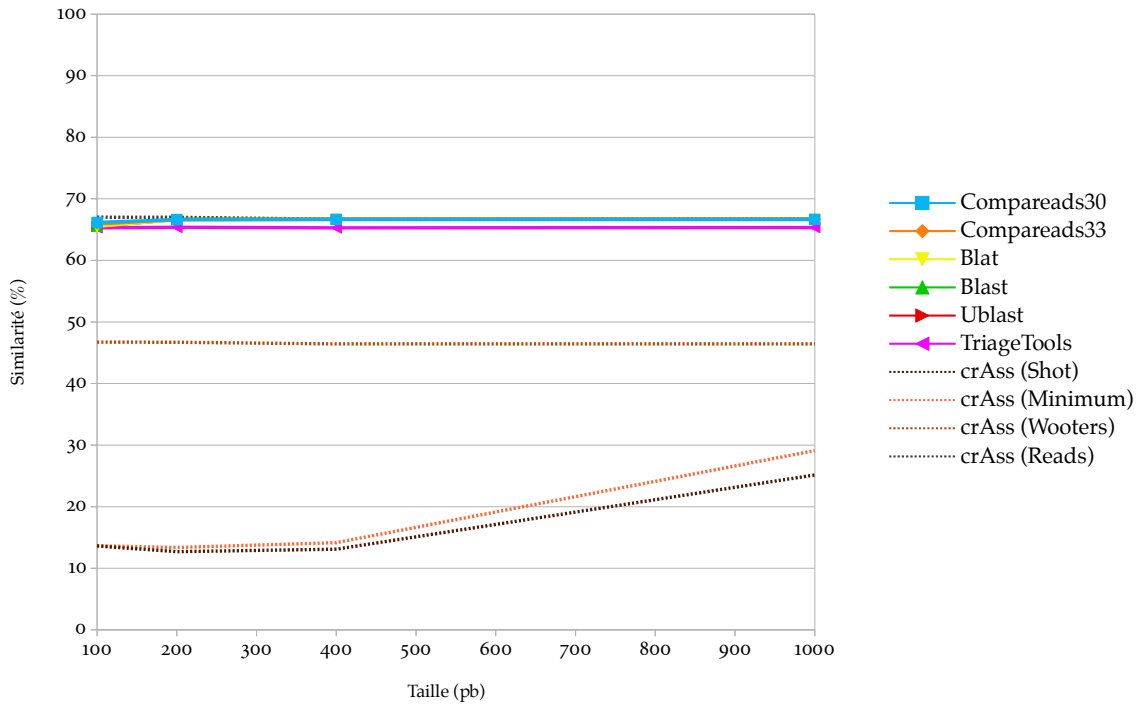


FIGURE 35: Représentation de la similarité trouvée par les différents logiciels en fonction de la taille des séquences, pour un nombre constant de paires de bases. Les jeux de données comportent trois génomes d'un million de paires de bases chacun et partagent deux tiers de ces génomes. Quatre notions différentes de similarité sont représentées pour CRASS".

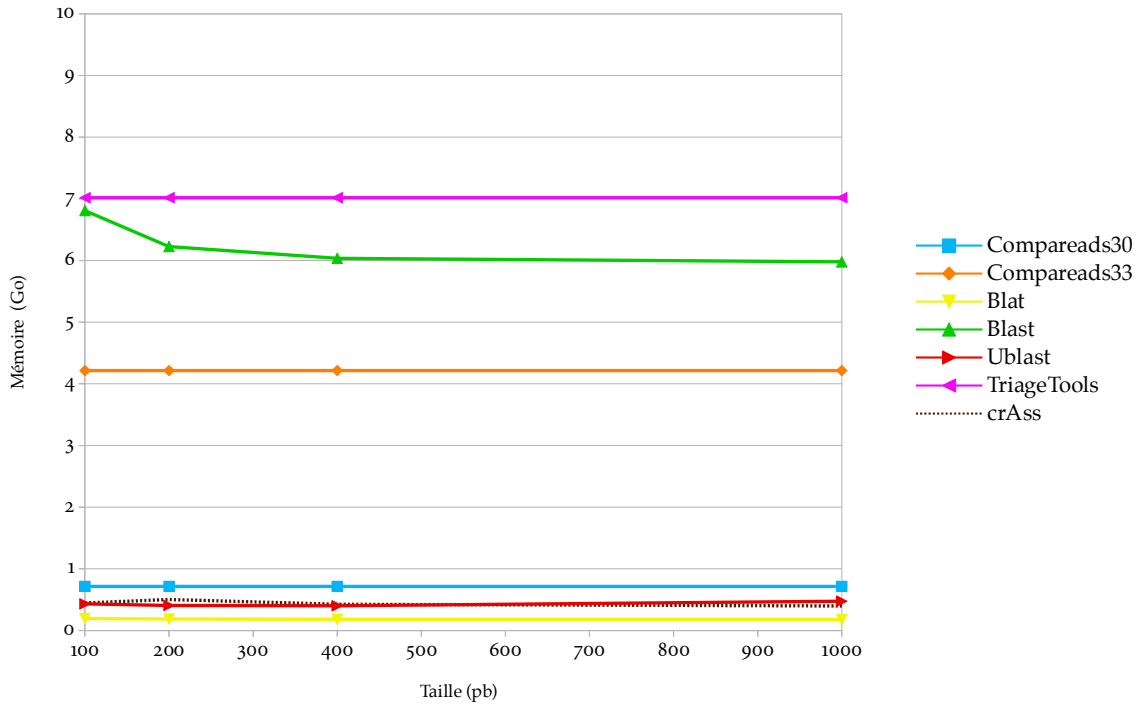


FIGURE 36: Représentation de l'utilisation mémoire des différents logiciels en fonction de la taille des séquences, pour un nombre constant de paires de bases. Les jeux de données comportent trois génomes d'un million de paires de bases chacun et partagent deux tiers de ces génomes.

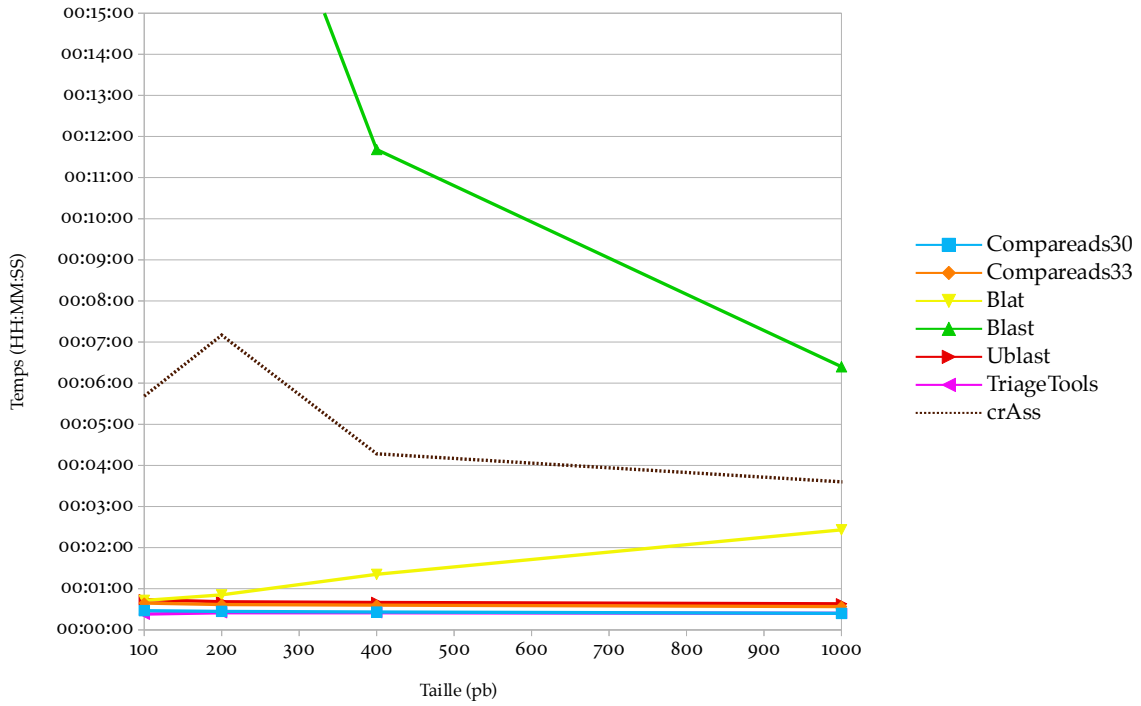


FIGURE 37: Représentation du temps d'exécution des différents logiciels en fonction de la taille des séquences, pour un nombre constant de paires de bases. Les jeux de données comportent trois génomes d'un million de paires de bases chacun et partagent deux tiers de ces génomes.

et ont une légère tendance à décroître : moins il y a de séquences, plus ils sont rapides. BLAST a le même comportement mais en bien plus exacerbé. Pour tous ces logiciels, ceci confirme l'étude sur l'évolution du nombre de séquences (voir 4.2.1.3). Pour BLAT, les résultats sont similaires à ceux sur l'évolution de la taille des séquences (voir 4.2.1.2) : BLAT est bien plus sensible à la taille des séquences qu'à leur nombre. Les approches les plus rapides sont UBLAST, TRIAGETOOLS et COMPAREADS.

Sur ce test, UBLAST et COMPAREADS sont les logiciels se comportant le mieux, tant en terme de mémoire, de temps d'exécution et de score de similarité.

4.2.2.2 *Évolution de la couverture des génomes*

La couverture des génomes dans un métagénome peut varier suivant les milieux étudiés et les technologies utilisées. Dans cette partie, on étudie le comportement des différents outils en fonction de la couverture des génomes. Pour cela, deux métagénomes sont créés, contenant chacun 3 génomes d'un million de paires de bases chacun. Les deux métagénomes partagent deux génomes. La couverture varie entre 1x et 10x et le nombre de séquences par jeu de données varie donc de 30 000 à 300 000 séquence de 100 pb.
La figure 38 montre les résultats de la similarité trouvée. On remarque que tous les logiciels ont du mal à retrouver la similarité réelle en dessous de 4x à 5x. Ceci indique que la faible couverture produit un sous échantillonnage aléatoire des génomes : il y a donc peu de séquences similaires entre deux jeux quand la couverture est faible.
On voit sur cette figure que CRASS est le seul logiciel dont les scores de similarité diminuent avec l'augmentation de la couverture. En utilisant CRASS, deux jeux à comparer sont tout d'abord assemblés ensemble. Avec une faible couverture, les séquences des génomes non partagés ont peu de chance d'être assemblées. Par contre, la couverture des génomes identiques dans les deux jeux est meilleure. Dès lors, la plupart des contigs créés lors de l'assemblage proviennent des génomes partagés. Les distances de CRASS se basent sur le nombre de sé-

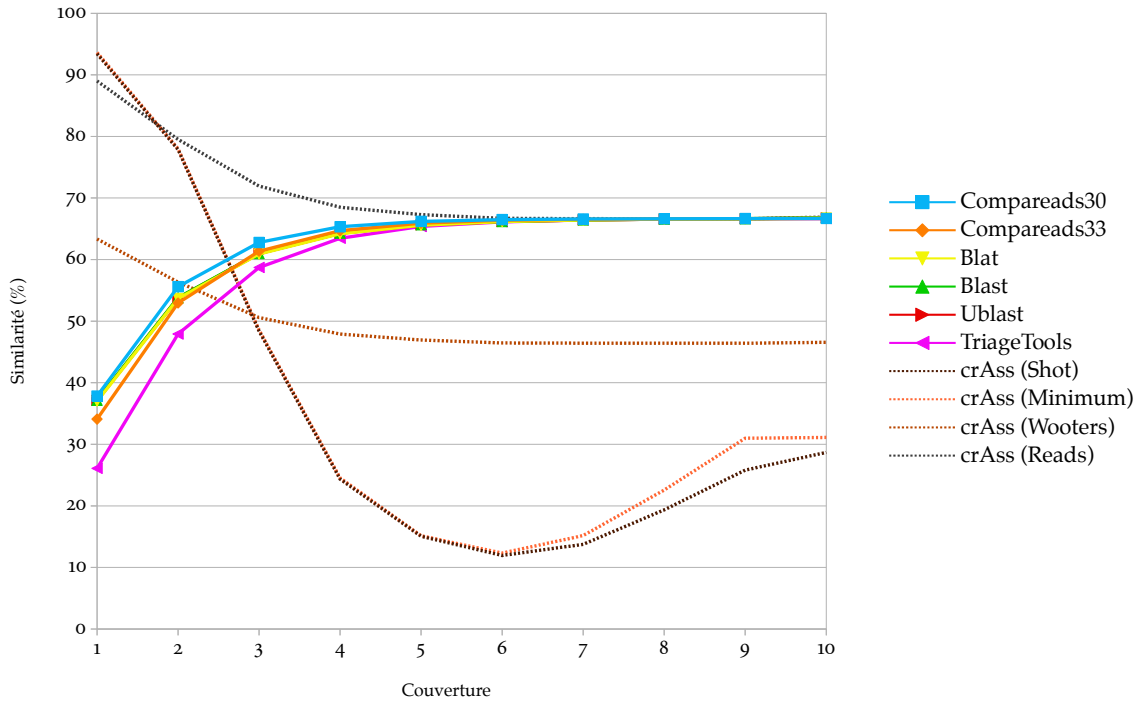


FIGURE 38: Représentation de la similarité trouvée par les différents logiciels en fonction de la couverture. Les jeux de données comportent trois génomes d'un million de paires de bases et partagent deux tiers de ces génomes. Quatre notions différentes de similarité sont représentées pour CRASS.

quences de chacun des métagénomes présentes dans les différents contigs. Or, comme les contigs viennent presque exclusivement des génomes partagés, CRASS en déduit que toutes les séquences sont similaires et surestime donc la similarité réelle. Cet effet se dissipe avec l'augmentation de la couverture : en effet, les génomes non-partagés ont alors plus de chance d'être assemblés et le calcul de distance est alors plus réaliste. Il est intéressant de noter qu'une fois encore, c'est la distance "Reads" qui est la plus fidèle à la similarité réelle.

La figure 39 confirme que le temps d'exécution des logiciels croît avec la quantité de séquence à traiter. TRIAGETOOLS et COMPAREADS ont les croissances les plus faibles. UBLAST et BLAT ont une croissance similaire, plus importante. Finalement, BLAST et CRASS sont à la fois les méthodes les plus lentes et dont l'augmentation du temps de calcul en fonction de la quantité de données à traiter est la plus grande. L'utilisation mémoire des logiciels est similaire aux autres expériences et n'est donc pas montrée.

Pour tous les logiciels, une faible couverture ne permet pas de trouver la similarité réelle ; il faut une couverture d'au moins 4x à 5x pour avoir un résultat proche de la similarité réelle. TRIAGETOOLS et COMPAREADS sont les solutions les plus intéressantes en terme de temps de calcul et de mémoire.

4.2.2.3 *Évolution du nombre de génomes*

Dans cette partie, on étudie le comportement des différents outils en fonction du nombre de génomes dans les métagénomes. Pour chaque tests, deux jeux de données sont créés. Pour un test donné, les deux échantillons comportent chacun le même nombre de génomes d'un million de paires de bases. Le séquençage se fait à 5x et génère des séquences de 100 pb. Le premier test concerne un unique génome, identique dans les deux jeux de données. Dans le second test, on utilise trois génomes dans chaque jeux, dont deux sont partagés. Dans le troisième test, on utilise cinq génomes dans chaque jeux, dont quatre sont partagés. Dans le

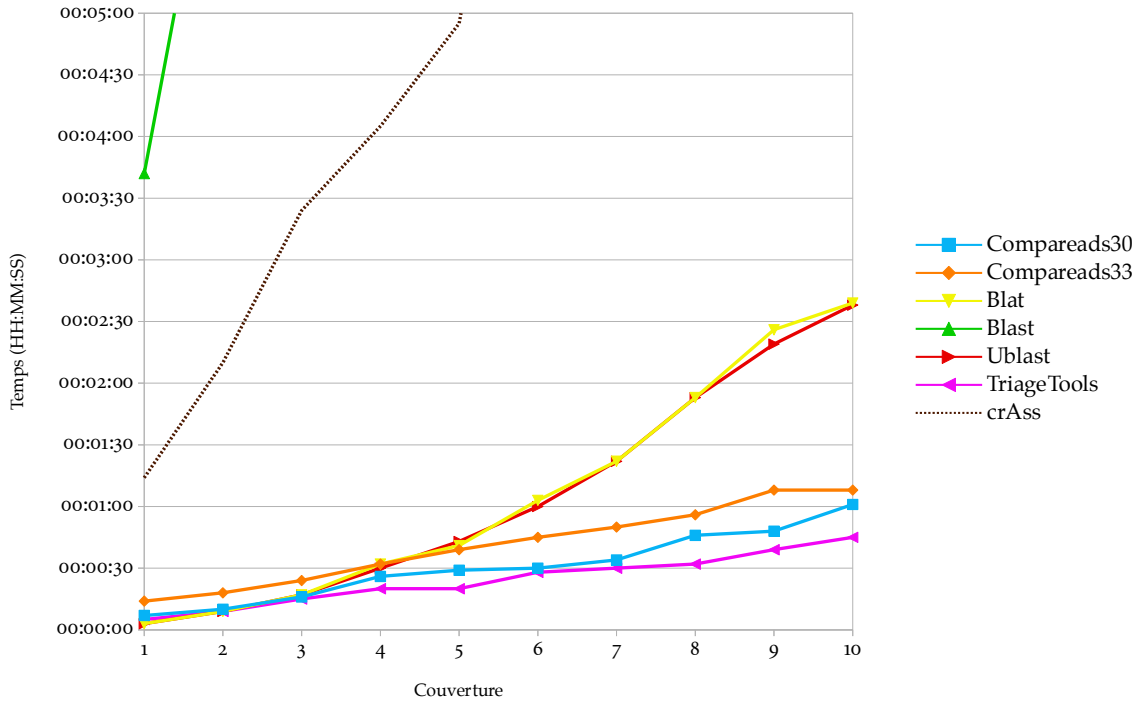


FIGURE 39: Représentation du temps d'exécution des différents logiciels en fonction de la couverture. Les jeux de données comportent trois génomes d'un million de paires de bases et partagent deux tiers de ces génomes.

quatrième test, on utilise sept génomes dans chaque jeux, dont cinq sont partagés. Dans le cinquième test, on utilise neuf génomes dans chaque jeux, dont sept sont partagés. La similarité réelle des jeux lors des cinq tests est donc, respectivement, 100%, 66,67%, 80%, 71,43% et 77,78%.

La figure 40 montre les résultats de la similarité trouvée. Tous les logiciels retrouvent la similarité réelle, et, une fois encore pour CRASS, c'est la distance "READS" qui est la plus proche de la similarité réelle.

La figure 41 confirme que le temps d'exécution des logiciels croît avec la quantité de séquence à traiter. TRIAGETOOLS et COMPAREADS ont les croissances les plus faibles. UBLAST et BLAT ont une croissance similaire, plus importante. Finalement, BLAST et CRASS sont à la fois les méthodes les plus lentes et dont l'augmentation du temps de calcul en fonction de la quantité de données à traiter est la plus grande. L'utilisation mémoire des logiciels est similaire aux autres expériences et n'est donc pas montrée.

L'augmentation du nombre d'espèces est similaire à l'augmentation du nombre de séquences. Tous les logiciels arrivent à retrouver la similarité réelle. TRIAGETOOLS et COMPAREADS sont les solutions les plus intéressantes en terme de temps de calcul et de mémoire.

4.2.2.4 *Évolution du nombre de substitutions*

Dans cette partie, on étudie le comportement des différents outils en fonction du nombre de substitutions introduites dans les génomes. Il faut noter que les substitutions sont introduites avant séquençage. Le but n'est pas de mimer les erreurs de séquençage mais bien d'étudier les outils face à des espèces plus ou moins proches. Pour cela, deux métagénomes sont créés, contenant chacun 3 génomes d'un million de paires de bases chacun. Les deux métagénomes partagent deux génomes. Un des deux métagénomes comporte les génomes normaux et le second, les génomes ayant reçu X substitutions. Le séquençage se fait à 5x et génère des séquences de 100 pb.

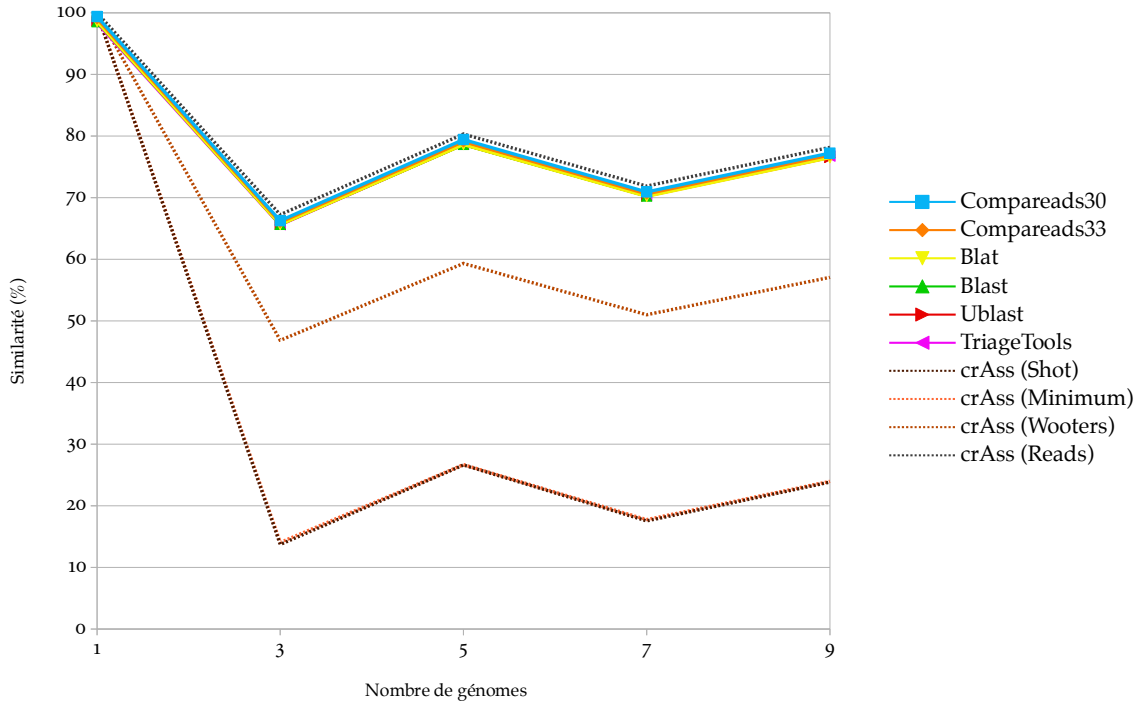


FIGURE 40: Représentation de la similarité trouvée par les différents logiciels en fonction du nombre de génomes. Les métagénomes sont séquencés à 5x et sont composés de séquences de 100 pb. Quatre notions différentes de similarité sont représentées pour crass.

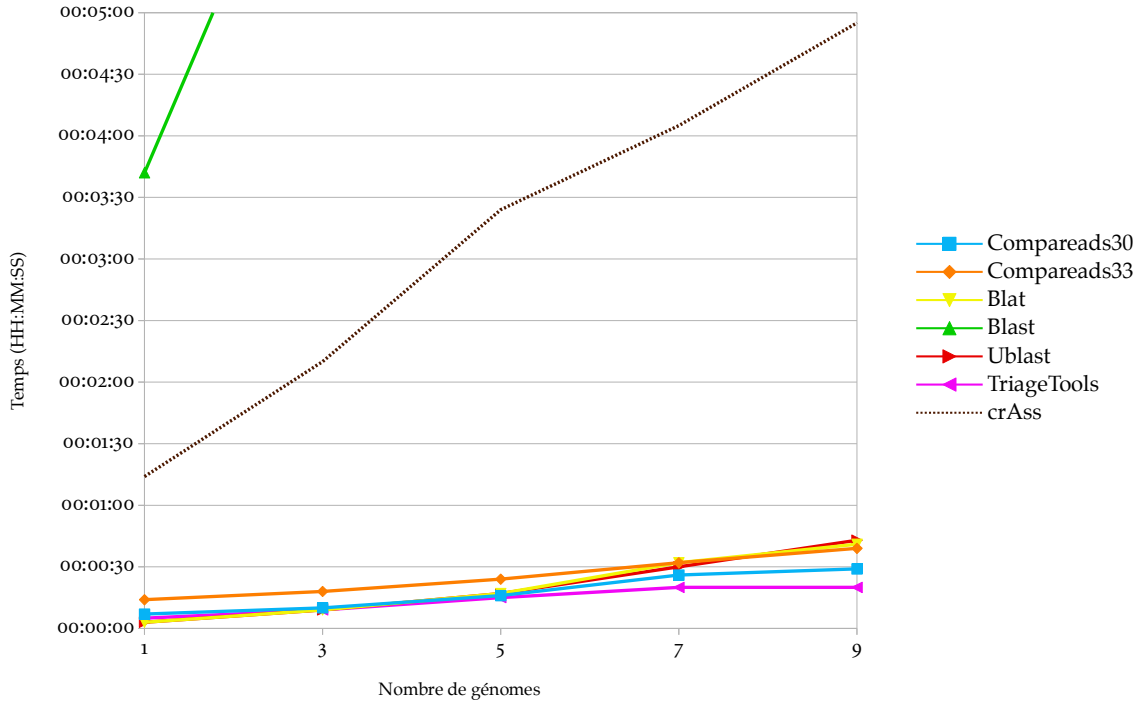


FIGURE 41: Représentation du temps d'exécution des différents logiciels fonction du nombre de génomes que contiennent les métagénomes. Les métagénomes sont séquencés à 5x et sont composés de séquences de 100 pb.

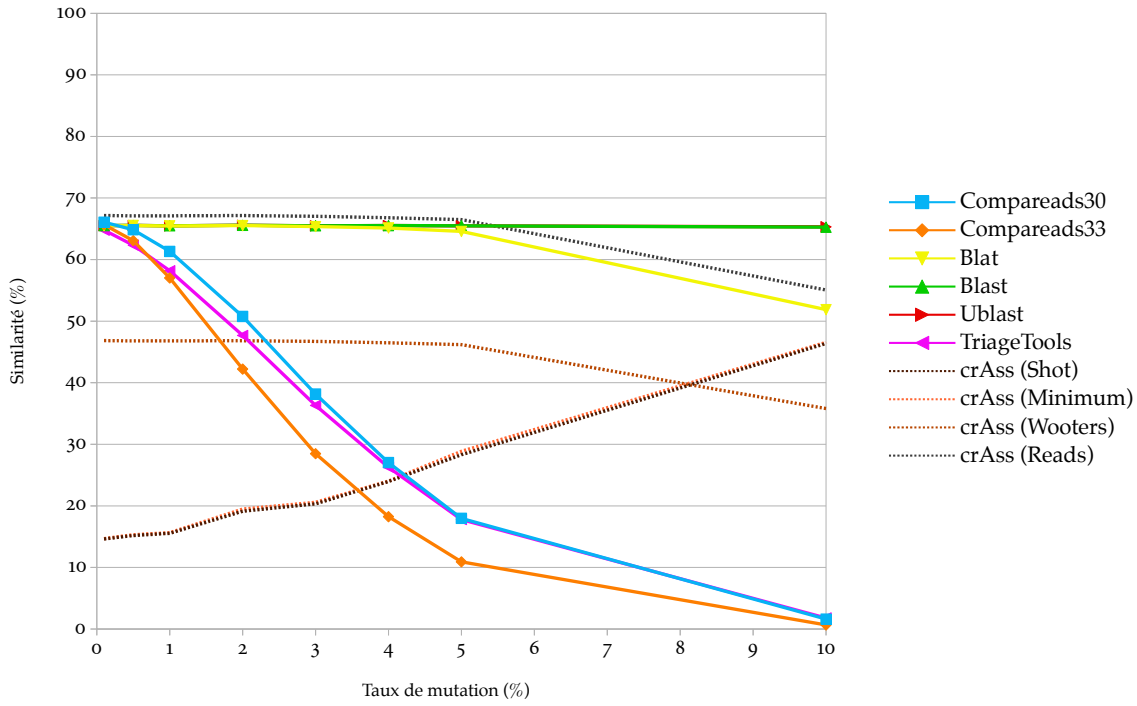


FIGURE 42: Représentation de la similarité trouvée par les différents logiciels en fonction du taux de mutation des génomes. Les métagénomes sont séquencés à 5x et sont composés de séquences de 100 pb. Les approches basées sur les *k*-mers (TRIAGETOOLS et COMPAREADS) chutent rapidement, quand les méthodes d'alignement (BLAT, UBLAST et BLAST) arrivent à garder un taux de similarité assez réaliste.

La figure 42 montre les résultats de la similarité trouvée. On peut remarquer que les approches basées sur les *k*-mers partagés, donc TRIAGETOOLS et COMPAREADS, perdent rapidement en sensibilité. Jusqu'à 0,5%, ils restent relativement bon : Compareads30 est à 64,8% au lieu de 66,6%, Compareads33 à 63,1% et TRIAGETOOLS à 62,4%. Les approches basées sur l'alignement (BLAST, BLAT et UBLAST) sont à 65,5% et CRASS à 67,2% pour la distance "READS". À 1% de mutations, les logiciels d'alignement sont toujours à 65,5% quand Compareads33 est à 57,0% et TRIAGETOOLS à 58,2%. À 10% de mutations, BLAST et UBLAST sont toujours à plus de 65% de séquences similaires quand Compareads33 est à 0,7% et TRIAGETOOLS à 1,8%. BLAT et CRASS (distance "READS") suivent une même décroissance pour terminer respectivement à 51,9% et 55,1%. Ceci n'est pas étonnant, COMPAREADS recherche des *k*-mers exacts dans le but d'identifier des séquences presque identiques. D'un point de vue biologique, un petit nombre de mutations entre deux génomes peut constituer deux espèces différentes. Dès lors, en terme de métagénomique comparative *de novo*, il est important de pouvoir différencier deux séquences différentes, même à quelques mutations près.

L'utilisation mémoire des logiciels et leur temps de calcul sont similaires aux autres expériences et ne sont donc pas montrés.

Le taux de mutation influe énormément sur TRIAGETOOLS et COMPAREADS, mais très peu sur les approches d'alignement. Ceci est un des plus gros défauts de COMPAREADS : l'utilisation de grands *k*-mers partagés ne permet pas de trouver des séquences similaires contenant plus de quelques pour-cents de substitution. Cependant, en pratique, l'utilisation de COMPAREADS permet d'obtenir des résultats concordant avec d'autre techniques (voir sous-section 4.2). De plus, le score de similarité peut être utilisé de manière relative : ce score peut alors servir à comparer entre eux plus de deux échantillons (voir sous-section 4.3).

CONCLUSION L'ensemble de ces résultats montre que TriageTools et CompareADS sont très comparables sur des petits jeux de données simulées. TriageTools est plus rapide sur des gros jeux de données, mais demande deux fois plus de mémoire et a un taux de faux positifs très élevé quand la quantité de données à traiter en importante. Dans la sous-section suivante, on compare ces deux outils sur un jeu de données réelles.

4.2.3 Performances sur un jeu de données réelles

Seuls TriageTools et CompareADS sont capables de traiter deux métagénomes de plus d'une centaine de millions de séquences (voir sous-sous-section 4.2.1.2). Ici, ces deux logiciels sont comparés sur deux métagénomes réels provenant de l'expédition Tara Oceans. Le premier échantillon comporte 165 235 024 séquences de 100 pb et le second, 169 344 074 séquences de 100 pb. CompareADS a été utilisé avec $k = 33$ et $t = 2$: deux séquences sont similaires si elles partagent au moins 2 33-mers non chevauchants. Il a terminé le calcul en 16h05m59s, en utilisant 4,3 Go de mémoire et a trouvé 13,58% de séquences similaires entre les deux jeux. TriageTools, avec les paramètres par défaut ($k = 14$ et $H = 36$), a mis environ quatre fois moins de temps, soit 3h58m52s mais a utilisé presque deux fois plus de mémoire, 7,0 Go. Le score de similarité qu'il obtient est de 99,48%. CompareADS étant une surestimation du "vrai" score de similarité, il est clair que TriageTools n'est pas capable de traiter ce gros échantillon métagénomique. En appliquant l'équation 1 avec les paramètres par défaut, le taux de faux positif est de 100%. Il est à noter que selon cette équation, TriageTools peut calculer l'intersection de deux échantillons de 150 millions de séquences de 100 pb en utilisant $k = 20$. Son taux de faux positif est alors de 0,81% et la quantité de mémoire nécessaire seulement pour sa structure de données est de 128 Go. L'implémentation actuelle de TriageTools ne supporte pas de valeur de k supérieures à 15.

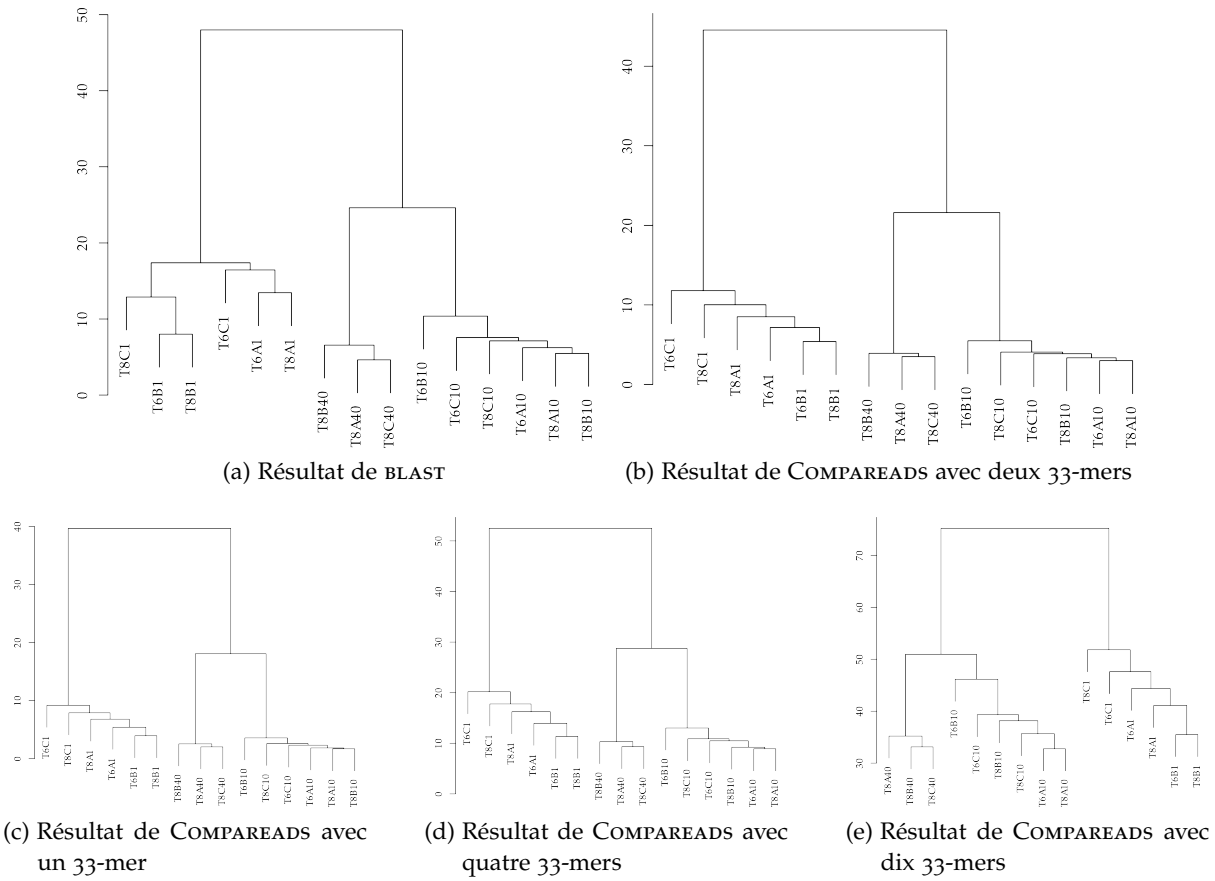
CONCLUSION Tous les logiciels testés dans la section précédente permettent de trouver un score de similarité entre deux ensembles de séquences. En terme de métagénomique comparative, seul CompareADS est capable de traiter les échantillons actuels, composés de centaines de millions de séquences. En outre, il permet d'identifier, et donc de récupérer, les séquences similaires entre les différents échantillons, chose qu'une approche comme CRASS ne permet pas de faire directement. Finalement, il utilise une quantité de mémoire faible et constante.

4.3 RÉSULTATS DE COMPAREADS SUR DES DONNÉES MÉTAGÉNOMIQUES RÉELLES

Dans cette section, on utilise CompareADS sur différents projets de métagénomique. Pour chaque projet, tous les échantillons ont été comparés deux à deux en utilisant un certain nombre de k-mers de 33 nucléotides. Lorsque plus de deux jeux doivent être comparés, toutes les comparaisons deux à deux sont effectuées. Ainsi, n jeux de données sont analysés en effectuant $(n-1) + (n-2) + ... + 1 = \frac{n^2-n}{2}$ comparaisons.

4.3.1 Étude de dénitrification

Une étude non publiée (*pers. comm.* Philippe Vandenkoornhuyse & Alexis Dufresne), sur la dénitrification d'eaux douces par des bactéries, a conduit à l'obtention de 15 métagénomes prélevés dans 3 conditions différentes. Chaque condition correspond à un ratio donné de carbone



(a) Résultat de BLAST

(b) Résultat de COMPAREADS avec deux 33-mers

(c) Résultat de COMPAREADS avec un 33-mer

(d) Résultat de COMPAREADS avec quatre 33-mers

(e) Résultat de COMPAREADS avec dix 33-mers

FIGURE 43: Résultats des intersections deux à deux de BLAST et de différentes versions de COMPAREADS sur des petits jeux réels de données métagénomiques. BLAST a été utilisé pour trouver les séquences s'alignant sur au moins 80 nucléotides avec une identité de séquence d'au moins 90%. Quatre expériences ont été conduites avec COMPAREADS afin d'identifier les séquences similaires sur, respectivement, 1, 2, 4 et 10 33-mers.

/ azote. Ces métagénomes comportent 3 réplicats techniques, identifiés par les lettres A, B et C. Les échantillons contiennent entre 104 252 et 282 664 séquences, avec une moyenne de 176 409 séquences par échantillon. Les séquences ont une longueur moyenne de 399 nucléotides (technologie ROCHE 454). Pour analyser les 15 jeux les uns contres les autre, 105 comparaisons ont été effectuées.

BLAST et COMPAREADS ont été utilisés pour trouver les séquences similaires sur les 105 comparaisons entre les 15 jeux. BLAST a été configuré pour trouver un alignement entre deux séquences d'au moins 80 nucléotides avec plus de 90% d'identité de séquence. Quatre expérimentations ont été conduites avec COMPAREADS afin de calculer, respectivement, les séquences similaires sur 1, 2, 4 et 10 33-mers. Pour chacune des cinq expériences, une clusterisation hiérarchique a été réalisée sur la base du score de similarité, donc du nombre de séquences similaires, entre tous les jeux, pris deux à deux. La figure 43 représente les dendrogrammes issus de ce clustering pour les cinq résultats.

Comme on peut le voir sur cette figure, les résultats de BLAST (43a) sont légèrement différents de ceux de COMPAREADS (43b), mais les trois branches principales sont les mêmes. Ces trois branches comportent chacune les échantillons provenant de seulement une des trois conditions biologiques marquée 1, 10 ou 40. La condition 1 correspond à l'ajout de carbone dans le milieu, 10 correspond à des conditions normales et 40 correspond à l'ajout d'azote dans le milieu.

De même, on voit figures 43c, 43b, 43d et 43e que COMPAREADS arrive à retrouver ces branches, même en effectuant une recherche très stricte. En effet, la figure 43e consiste à

trouver des séquences similaires sur 10 33-mers distincts, soit 330 nucléotides sur les 399 en moyenne. Bien que moins de séquences soient alors considérées comme similaires, le score relatif est capable de discriminer les trois conditions.

Ces jeux de données montrent la cohérence des résultats entre BLAST et COMPAREADS. De plus, on voit ici la robustesse de COMPAREADS et sa pertinence à retrouver les trois conditions.

4.3.2 Métagénomique intestinale de l'escargot

Dans une récente étude sur le métagénome de l'escargot *Achatina fulica* [98], Ana Tereza R. Vasconcelos *et al.* ont comparé 34 métagénomes provenant d'IMG/M [72] avec le métagénome intestinal d'*Achatina fulica*. La comparaison se base sur les similarités enzymatiques produites par les métagénomes, en interrogeant la base de données CAZY. CAZY est une base de données d'enzymes, classées par familles de structures protéiques. Le résultat de cette comparaison a été mis en forme par clustering hiérarchique et est représenté sur le dendrogramme figure 44.

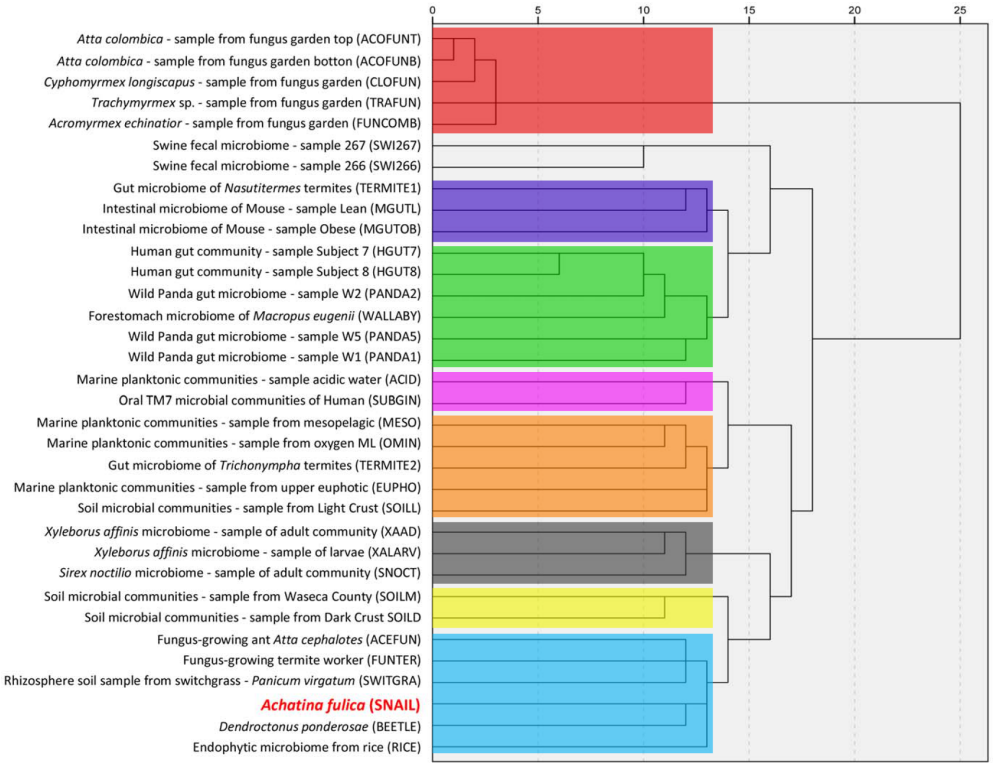


FIGURE 44: Dendrogramme représentant la comparaison enzymatique de 34 métagénomes issue de la publication d'Ana Tereza R. Vasconcelos *et al.* [98]. Le dendrogramme est réalisé suite à un clustering hiérarchique. Les couleurs permettent de différencier des clusters. Les clusters contenant les deux métagénomes de porc n'est pas coloré.

Les deux métagénomes de porc (SWI266 et SWI267) n'étant plus disponibles sur IMG/M, 32 des 34 métagénomes ont été comparés deux à deux avec COMPAREADS, en recherchant 2 33-mers de partagés entre toutes les séquences. Les résultats sont présentés, après clusterisation hiérarchique, figure 45. On peut remarquer que tous les échantillons de fonges (ACOFUNB, ACOFUNT, ACEFUN, CLOFUN, FUNCOMB, FUNTER et TRAFUN) sont regroupés ensemble par COMPAREADS alors que cinq sur les sept sont ensemble dans la publication d'origine. Dans les deux études, deux des trois échantillons de sol sont ensemble, mais ce ne sont pas les deux mêmes. Dans la publication d'origine, les métagénomes intestinaux d'humains, de pandas

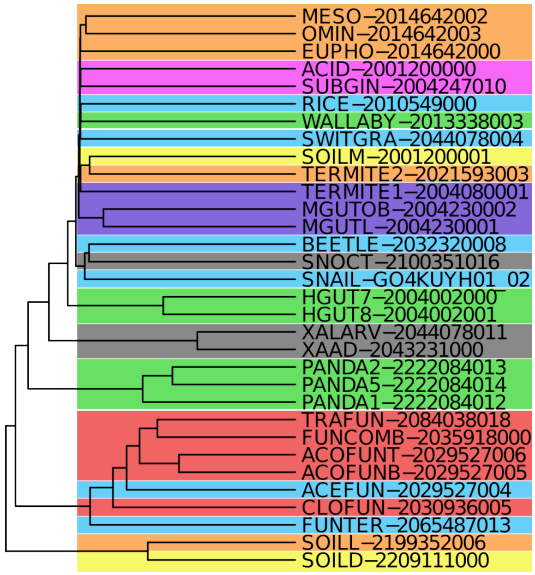


FIGURE 45: Dendrogramme représentant le résultat de Compareads sur 32 métagénomes. Le dendrogramme est réalisé suite à un clustering hiérarchique. Les couleurs permettent de différencier les clusters identifiés figure 44.

et de wallaby sont regroupés ensemble. Compareads sépare les humains des pandas et le wallaby n'apporte pas assez de signal pour être correctement clusterisé.

Deux autres clusters sont intéressants. Le premier, dans COMPAREADS, regroupe les deux échantillons provenant d'insectes du genre *Xyleborus*. Ces deux insectes sont corrélés, dans la publication de base, avec un autre arthropode, *Sirex noctilio*. Ce dernier insecte est, avec COMPAREADS, dans le second cluster d'intérêt, cluster qui regroupe les échantillons d'escargot et de deux arthropodes, une espèce de coccinelle et *Sirex noctilio*. Dans l'étude d'origine, l'escargot et la coccinelle sont regroupés avec le riz, deux fonges et un échantillon de sol.

Les différences entre les deux approches sont significatives, mais il faut garder en tête que les deux méthodes utilisées sont très différentes. De plus, on ne dispose pas d'une référence permettant de juger laquelle des deux méthodes est la plus proche de la vérité ; d'un point de vue général, les deux approches apportent de l'information, et semblent complémentaires. D'un point de vue performance, le temps moyen d'une intersection entre deux métagénomes avec COMPAREADS est de 2m07s : ces métagénomes sont petits, ils contiennent en moyenne 152 050 séquences de quelques centaines de paires de bases.

4.3.3 Metasoil

Comme expliqué sous-section 2.2.2, l'étude des métagénomes de METASOIL a montré que les échantillons traités par une même technique d'extraction partagent plus de similarité au niveau fonction que deux échantillons traités par des techniques d'extractions différentes [120]. Pour rappel, les treize métagénomes de METASOIL, deux autres métagénomes de sol et un métagénome d'eau de mer de GOS ont été clusterisés sur la base du nombre de fonctions de références qu'ils partagent les uns avec les autres, à l'aide de MG-RAST. Le dendrogramme issu de cette comparaison est montré figure 4.3.3. Tous ces métagénomes, sauf un provenant de sol italien non disponible, ont été analysés les uns contre les autres avec COMPAREADS. Le dendrogramme issu de COMPAREADS (voir figure 47) est très proche de celui de base.

Sur ces deux figures, tous les échantillons provenant d'extraction directe d'ADN sont regroupés ensemble. Sur la figure provenant de COMPAREADS, on voit que tous les échantillons provenant d'extraction indirecte sont aussi regroupés ensemble, ce qui n'est pas le cas sur

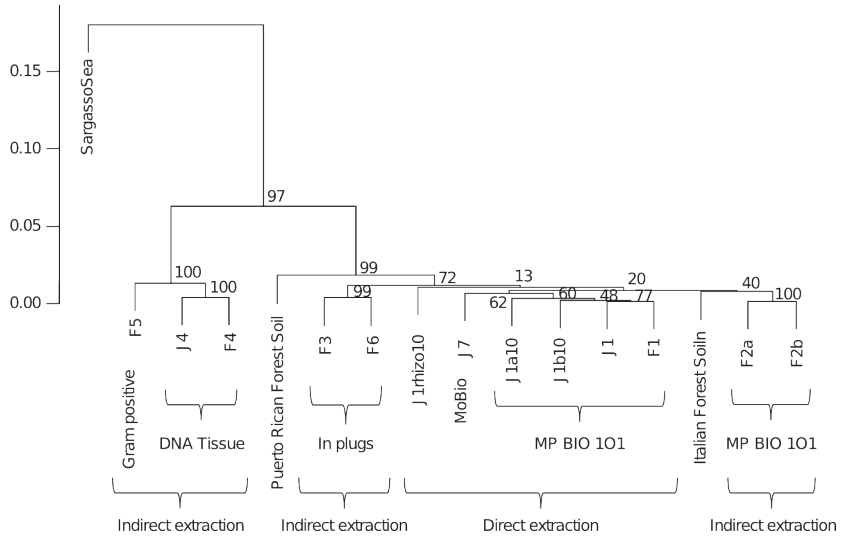


FIGURE 46: Dendrogramme obtenu sur quinze métagénomes, dont les treize de metasoil. La comparaison a été menée avec MG-RAST [120]. Le calcul de similarité entre les échantillons est basé sur 835 fonctions, chacune présente dans au moins un des échantillons.

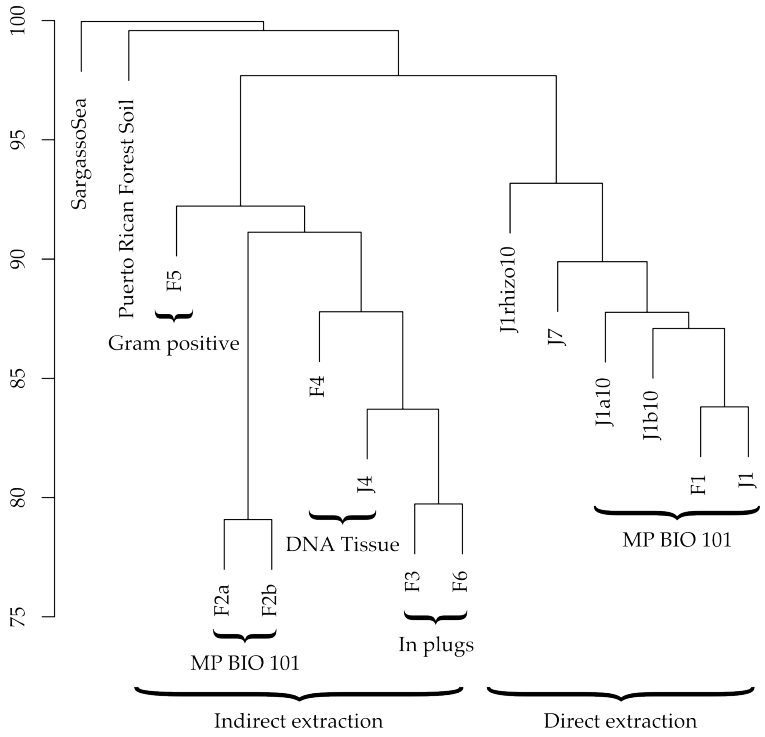


FIGURE 47: Dendrogramme obtenu sur quatorze métagénomes, dont les treize de metasoil. La comparaison a été menée avec COMPAREADS en utilisant 2 33-mers partagés pour identifier les séquences similaires entre deux échantillons.

la figure d'origine. En allant plus loin, on voit que les sous-groupes sont très proches dans les deux cas. Les groupes "*In plugs lysis*", "MP BIO *101*" et "*Gram positive*" sont similaires. Le groupe "DNA *tissue*" est mieux représenté sur la figure d'origine. Finalement, dans la figure de COMPAREADS, les deux échantillons ne provenant pas de Metasoil sont tout à fait à l'extérieur de l'arbre. Dans la publication d'origine, l'échantillon d'eau de mer est très loin des autres, mais le sol de Puerto Rico semble beaucoup plus proche. Malgré cette proximité, cet échantillon reste loin de tous les autres.

Cette analyse confirme la pertinence biologique de COMPAREADS. Bien que fondamentalement différentes, les deux approches mènent à la même conclusion, à savoir que les techniques d'extraction d'ADN avant séquençage ont un impact réel sur les données obtenues. Ainsi, au sein d'une expérience, il est important d'utiliser une seule et unique technique d'extraction d'ADN.

4.3.4 Global ocean sampling

Dans une des publications de GOS [94] (voir sous-section 2.2.3), une analyse des métagénomes a été conduite sur la seule base des séquences similaires. La figure 7 page 34, tirée de cette publication, représente ces similarités entre échantillons. Les 44 métagénomes ont été analysés avec COMPAREADS. Les échantillons contiennent, en moyenne, 174 759 séquences de 1 249 nucléotides en moyenne (technologie sanger). les 990 intersections ont été calculées avec COMPAREADS ($t = 4$ et $k = 33$) en 72 heures et 30 minutes. En moyenne, une intersection a pris 4 minutes et 23 secondes. Dans la publication d'origine, aucune information de temps ou de mémoire n'est présentée.

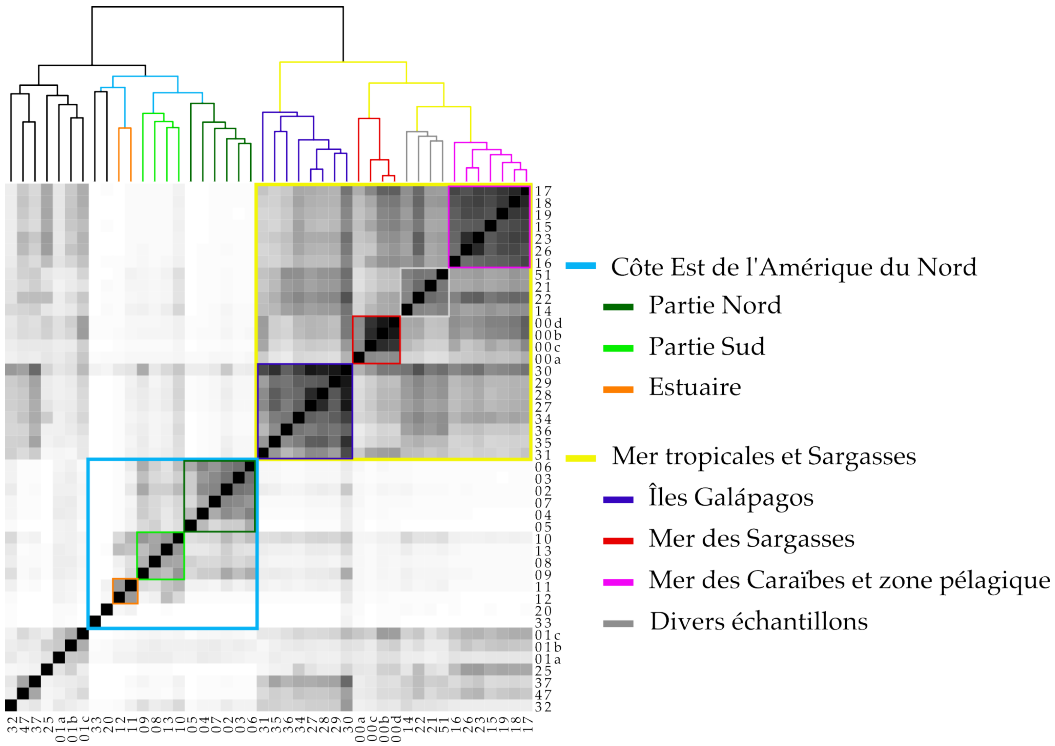


FIGURE 48: Représentation de la similarité entre les 44 échantillons de GOS obtenu avec COMPAREADS en recherchant 4 33-mers partagés entre chaque séquence. Les intersections en noir représentent un score de similarité de plus de 50% et les intersections en blanc, 0%.

Les résultats, figure 48 et figure 7, sont très proches entre les deux études. Deux groupes principaux sont bien établis. Le premier, représenté en turquoise, regroupe presque tous les échantillons venant d'eaux tempérées de la côte Est de l'Amérique du Nord. Seul l'échan-

tillon 14, comme dans la publication d'origine, n'est pas inclus dans cet ensemble. Ce groupe contient aussi deux échantillons très différents de tous les autres. Le premier provient d'eau douce et le second d'eau hypersaline. On peut diviser ce premier groupe principal en trois sous-parties. La première, en vert foncé, regroupe les échantillons provenant de la partie Nord des États-Unis et le groupe en vert clair, de la partie sud. Le troisième groupe, en orange, contient les deux échantillons venant d'estuaires. Ces trois groupes sont identiques à ceux de la publication d'origine.

Le second groupe principal coloré en jaune, contient les échantillons tropicaux et ceux de la mer des Sargasses. La sous-partie en violet foncé regroupe des échantillons provenant exclusivement des Galápagos. La sous-partie rouge contient les échantillons de la mer des Sargasses. Dans la publication d'origine, l'échantillon 00a n'est pas dans ce groupe. Selon les métadonnées, le groupe en gris, comme dans la publication d'origine, contient des échantillons venant de plusieurs endroits. Finalement, le groupe en magenta contient des échantillons de zones pélagiques et des Caraïbes, comme dans la publication d'origine.

Ces résultats montrent que Compareads est capable de classer de nombreux métagénomes en fonction de leurs lieux de prélèvements, en utilisant seulement des informations de séquence. La provenance géographique générale (tropique ou Amérique du Nord) permet de séparer en deux les échantillons. La localisation plus précise (Galápagos, sud des États-Unis, etc) est aussi retrouvée.

4.3.5 Tara oceans

Parmi toutes les stations en mer effectuées lors de l'expédition Tara Oceans, une trentaine a été réalisée entre le sud de l'Afrique du sud, le haut de l'océan Atlantique sud et le sud de l'Amérique latine (voir figure 49). Ces prélèvements ont été réalisés pour étudier de grands systèmes océaniques.

Dans l'ouest de l'océan Indien, des courants viennent buter contre l'Afrique. Des eaux s'écoulent donc autour de Madagascar, vers le sud-ouest, et génèrent le courant des Aiguilles (Agulhas en portugais). Ce courant s'écoule le long de la côte est sud-africaine, vers le sud-ouest. Au large de l'Afrique du sud, le courant des Aiguilles rencontre des eaux plus froides en sens inverse, venant de l'océan Atlantique et de l'océan Austral. Cette rencontre crée de vastes tourbillons nommés "anneaux d'Agulhas". Ces tourbillons mesurent des centaines de kilomètres de diamètre et peuvent atteindre 4000 mètres de profondeur. Les anneaux d'Agulhas sont composés d'eaux provenant essentiellement du canal du Mozambique et donc différentes, en température et en salinité, du reste des eaux de l'océan Atlantique.

Certains anneaux d'Agulhas sont entraînés vers la côte ouest sud-africaine dans le gyre de l'Atlantique sud. Un gyre est un gigantesque tourbillon d'eau océanique qui est formé par un ensemble de courants marins. Le gyre de l'Atlantique sud est formé, au sud, par le courant circumpolaire antarctique de l'océan Austral, courant qui remonte le long de la côte ouest sud-africaine par le courant de Benguela. Au niveau de l'Angola, le courant de Benguela rencontre les eaux équatoriales et est entraîné vers le Brésil. Les eaux sont alors transportées par le courant Brésilien, vers le sud. Finalement, au sud de l'Amérique latine, le courant Brésilien rencontre le courant des Malouines et le courant circumpolaire antarctique.

D'un point de vue métagénomique, les stations dans le canal du Mozambique visent à en déterminer la composition en organismes (stations 52, 64 et 65, en orange sur la figure 49). Ensuite, l'expédition a échantillonné un tourbillon en formation au sud du Cap (station 66). Finalement, la goélette a suivi la course des anneaux d'Agulhas dans le gyre de l'Atlantique sud pour déterminer à quel point ces tourbillons influent sur la circulation des microorganismes. Le long de la côte ouest sud-africaine, les eaux froides du courant de Benguela pro-

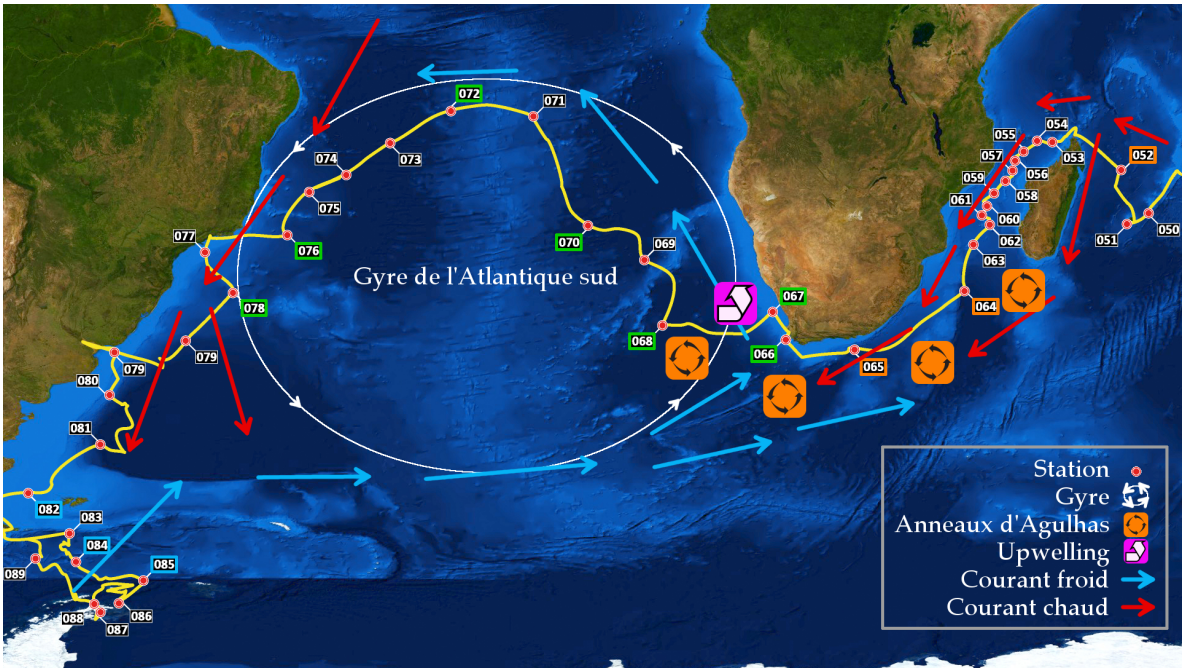


FIGURE 49: Prélèvements effectués au cours de l'expédition TARA OCEANS dans l'ATLANTIQUE SUD.

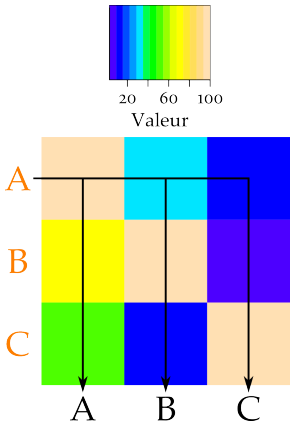


FIGURE 50: Sens de lecture des *heat maps* obtenues à partir du score de similarité entre trois échantillons A, B et C. Les résultats doivent être lus à partir de l'ordonnée.

duisent un phénomène nommé *upwelling* (remontée d'eau en français) : le long du courant du BENGUELA, des eaux profondes et froides remontent en surface et charrient de nombreux microorganismes. Lors de la station 67, a été effectué un prélèvement dans un site d'*upwelling* pour en étudier les différences, entre autres au niveau des organismes.

Les *heat maps* figures 51 et 52 représentent le pourcentage de similarité entre 13 échantillons provenant du canal du MOZAMBIQUE (légende en orange), du gyre de l'ATLANTIQUE SUD (légende en vert) et de l'océan AUSTRAL (légende en bleu). Pour certaines stations, il y a deux échantillons, l'un en surface (SUR) et le second à la profondeur où la chlorophylle est maximale (DCM pour *Deep Chlorophyll Maximum*).

L'ordre de lecture des *heat maps* obtenues à l'aide de COMPAREADS est important. L'intersection de deux échantillons A et B conduit à deux résultats : le pourcentage des séquences de A similaires à des séquences de B et le pourcentage des séquences de B similaires à des séquences de A.

La figure 50 montre le sens de lecture à suivre. Sur cette figure, 100% des séquences de A sont similaires à des séquences de A, 30% (en turquoise) des séquences de A sont similaires

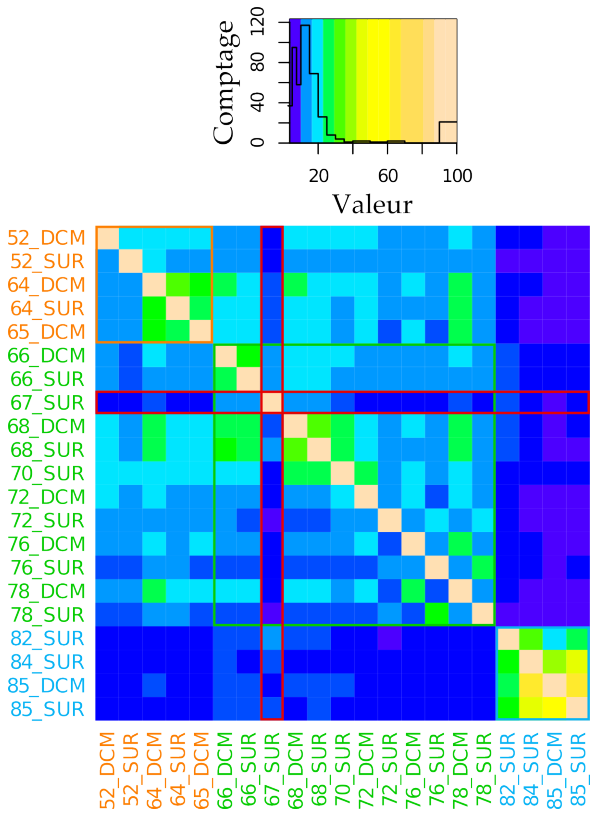


FIGURE 51: *Heat map* des intersections de 13 stations de TARA OCEANS pour des organismes d'une taille de 0,8 µm à 5 µm. Les stations dont la légende est en orange proviennent du canal du MOZAMBIQUE, les stations dont la légende est en vert du gyre de l'ATLANTIQUE sud et les stations dont la légende est en bleu, de l'océan AUSTRAL. La station 67 a été réalisée dans un site d'*upwelling*.

à des séquences de B et 10% des séquences de A sont similaires à des séquences de C (bleu foncé). De même 70% (en jaune) des séquences de B sont similaires à des séquences de A et moins de 10% des séquences de B sont similaires à des séquences de C. Finalement, environ 45% (en vert) des séquences de l'échantillon C sont similaires à des séquences de A et 10% à des séquences de B.

La *heat map* figure 51 a été obtenue avec des organismes d'une taille de 0,8 µm à 5 µm, contenant donc des bactéries et des petits eucaryotes. Sur cette figure, on voit que les échantillons provenant de l'océan AUSTRAL n'ont que très peu de séquences similaires avec les échantillons d'océan ATLANTIQUE ou INDIEN. Il semble ainsi que, pour les organismes de la taille étudiée, il n'y ait pas une frontière nette de répartition entre l'océan INDIEN et l'océan ATLANTIQUE. Par contre, l'océan AUSTRAL a un contenu en ADN très différent des deux autres océans étudiés.

Un autre point intéressant sur cette figure est la station 67 (encadrée en rouge), clairement différente de toutes les autres. Elle ne compte que très peu de séquences similaires avec les autres échantillons et semble donc contenir un écosystème différent. Pour rappel, le prélèvement a été réalisé dans un site d'*upwelling*. Il est connu que les sites d'*upwelling* sont particulièrement riches en organismes vivants, ce qui explique probablement une telle différence au niveau ADN entre cette station et les autres.

La *heat map* figure 52 a été obtenue avec des organismes d'une taille de 180 µm à 2000 µm, contenant donc des protistes, du zooplancton et d'autres eucaryotes. Comme sur la figure 51, les échantillons provenant de l'océan AUSTRAL n'ont que très peu de séquences similaires avec

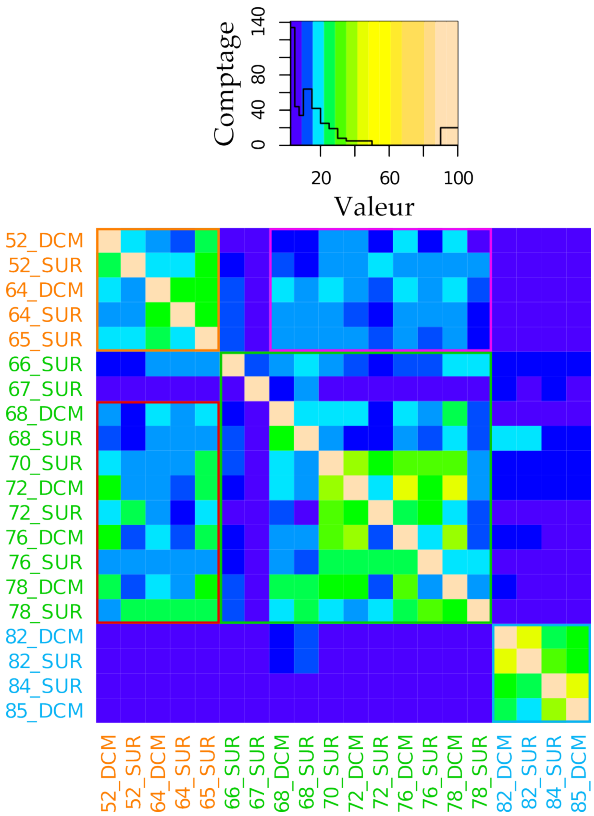


FIGURE 52: *Heat map* des intersections de 13 stations de TARA OCEANS pour des organismes d'une taille de 180 µm à 2000 µm. Les stations dont la légende est en orange proviennent du canal du MOZAMBIQUE, les stations dont la légende est en vert du gyre de l'ATLANTIQUE sud et les stations dont la légende est en bleu, de l'océan AUSTRAL. La station 67 a été réalisée dans un site d'*upwelling*. Le rectangle délimité en rouge représente le pourcentage de séquences de l'océan ATLANTIQUE sud similaires à des séquences de l'océan INDIEN. Le rectangle délimité en mauve représente le pourcentage de séquences de l'océan INDIEN similaires à des séquences de l'océan ATLANTIQUE sud.

les échantillons d'océan ATLANTIQUE ou INDIEN. De même la station 67 est clairement différente de toutes les autres stations.

L'observation la plus intéressante que l'on peut faire sur cette figure est délimitée par le rectangle rouge et le rectangle mauve. Le rectangle rouge représente le pourcentage de séquences de l'océan ATLANTIQUE sud similaires à des séquences de l'océan INDIEN et le rectangle mauve représente le pourcentage de séquences de l'océan INDIEN similaires à des séquences de l'océan ATLANTIQUE sud. On constate que le rectangle mauve contient 7 résultats supérieurs à 15% de similarité, mais aucun supérieur à 20%. Le rectangle rouge contient 19 valeurs supérieures à 15% dont 12 supérieures à 20% de similarité. Dès lors, on peut dire que pour la taille étudiée, une partie des séquences de l'océan ATLANTIQUE sud est similaire à des séquences de l'océan INDIEN, mais une moindre partie de séquences de l'océan INDIEN est similaire à des séquences de l'ATLANTIQUE sud. Une partie de l'ADN du zooplancton et des protistes de l'océan INDIEN est donc retrouvée dans l'océan ATLANTIQUE sud, mais l'inverse est moins fréquent. Cet ADN est peut-être transporté, depuis le MOZAMBIQUE vers l'ATLANTIQUE sud, par les anneaux d'AGULHAS.

Ces premiers résultats de COMPAREADS sur les métagénomes de TARA OCEANS semblent indiquer qu'un mélange s'opère entre les eaux de l'océan INDIEN et de l'océan ATLANTIQUE, peut-être au travers des anneaux d'AGULHAS. Ces résultats montrent ainsi que les grands systèmes océaniques influent sur la répartition globale des microorganismes marins. Si ces premiers résultats permettent de retrouver dans les grandes lignes certains mouvements connus des

masses d'eaux océaniques, les différentes tailles d'organismes étudiées donnent des images différentes. Ces différences restent à interpréter dans le détail.

4.4 CONCLUSION

L'outil COMPAREADS utilise la nouvelle structure de données présentée chapitre 3. Il est capable de comparer deux jeux de données métagénomiques en utilisant une faible quantité de mémoire. Cette comparaison ne s'appuie que sur les données elles-mêmes ; les connaissances actuelles ne rentrent pas en compte et toutes les séquences des jeux de données sont ainsi utilisées. De plus, aucun autre outil n'est nécessaire en amont, COMPAREADS fonctionne directement sur des jeux de données bruts. La comparaison s'effectue sur la quantité de séquences similaires entre deux jeux et un score de similarité global est délivré. Ce score de similarité peut être utilisé de manière relative afin de comparer plusieurs échantillons métagénomiques les uns contre les autres. De plus, les séquences similaires sont extraites et il est possible de lancer d'autres analyses sur ces seules séquences.

COMPAREADS est heuristique et des faux positifs sont générés ; ceux-ci sont peu nombreux avec les paramètres par défaut. Mais, en utilisant plus de données ou moins de mémoire, notre logiciel procède à plusieurs phases d'indexation indépendantes. Plus ce nombre de phase d'indexation augmente, plus les faux positifs augmentent. Par exemple, pour indexer tous les 30-mers de 100 millions de séquences de 100 pb, COMPAREADS procède à 63 phases d'indexation et génère près de 1% de faux positifs. Pour réduire la quantité de faux positifs, il convient d'utiliser une plus grande quantité de mémoire. Pour la même quantité de donnée, utiliser des 33-mers et 4 Go de mémoire permet d'obtenir un taux de faux positifs inférieur à 0,1%. De la même manière, COMPAREADS pourrait traiter 1 milliards de séquences de 100 pb en utilisant 4 Go de mémoire. Il conduirait alors à générer environ 1% de faux positifs. En passant à des 36-mers, l'empreinte mémoire serait de 32 Go et le taux de faux positifs inférieur à 0,1%.

Le logiciel COMPAREADS est dédié à l'analyse de grands jeux de données métagénomiques comportant des centaines de millions de séquences. Il est, pour le moment, le seul outil capable de comparer de tels jeux. Notre outil produit des résultats robustes, biologiquement exploitables et en accord avec diverses méthodes fondamentalement différentes. Il est actuellement utilisé de manière intensive sur les échantillons provenant de l'expédition TARA OCEANS. Les premiers résultats montrent que la dynamique globale de certains océans semble jouer un rôle sur la dispersion de microorganismes à l'échelle de la planète.

5 CONCLUSIONS ET PERSPECTIVES

Les travaux de thèse présentés dans ce manuscrit apportent trois contributions au domaine de la métagénomique. La première contribution est d'ordre sémantique et propose une classification du paysage métagénomique en quatre familles. Ces quatre familles que sont la métagénomique ciblée, quantitative, fonctionnelle et comparative, décrivent les différentes approches actuellement utilisées en métagénomique. Cette classification se veut comme un effort de clarification du domaine jeune et en pleine expansion qu'est la métagénomique.

La seconde contribution est une méthodologie et un outil, Compareads, dédiés à la métagénomique comparative *de novo*. Cet outil compare deux jeux de données métagénomiques en identifiant les séquences similaires entre ces deux jeux. Deux scores de similarité sont alors calculés sur la base des séquences similaires trouvées.

La comparaison de grands échantillons métagénomiques est un problème est pleine expansion : de plus en plus de projets métagénomiques génèrent un très grand nombre de grands jeux de données. Notre méthode fonctionne sur des échantillons bruts et ne nécessite aucune autre donnée externe. Plusieurs utilisations possibles de notre méthode existent. Le premier cas survient quand on cherche à isoler les séquences communes, ou à l'inverse différentes, entre plusieurs échantillons pour ensuite ne travailler que sur ces séquences. Compareads sert alors à réduire le nombre de séquences sur lesquelles utiliser des processus longs et coûteux en ressources. En se focalisant uniquement sur les séquences potentiellement intéressantes, les temps de calculs sont ainsi diminués.

Le second cas d'utilisation de Compareads est la comparaison de plusieurs échantillons dans un but de clustering. Notre méthode propose deux scores de similarité, utilisant toutes les séquences des jeux de données, qui fonctionnent à la fois de manière absolue (*i.e.* ces deux jeux ont tel pourcentage de similarité entre eux) que de manière relative (*i.e.* ces deux jeux sont plus proches entre eux que de ce troisième). Une clusterisation hiérarchique basée sur ces scores fournit une information pertinente sur les liens métagénomiques des différents échantillons étudiés.

Le troisième cas d'utilisation de Compareads se présente quand on dispose de métadonnées, par exemple physico-chimiques, sur les échantillons. On peut alors réaliser la comparaison de plusieurs échantillons à l'aide de Compareads puis injecter les données environnementales (température, salinité, pH, présence d'un élément, etc) dans les résultats pour y rechercher des corrélations. C'est actuellement ce processus qui est utilisé dans le projet tara oceans.

La troisième contribution est une nouvelle structure de données basée sur le filtre de bloom. Cette structure sert à indexer une grande quantité de k-mers (avec par exemple $k = 33$) puis à tester dans cet index la présence d'un k-mer donné. Les principaux avantages de cette structure sont sa rapidité, tant lors de la phase d'indexation que lors de la phase de recherche, et sa faible empreinte mémoire. Cette structure indexe et effectue une recherche dans l'index un ordre de grandeur plus rapidement que le filtre de bloom. Néanmoins, elle nécessite 2,5 fois plus de mémoire qu'un filtre de bloom traditionnel pour indexer la même quantité de données, à taux de faux positifs identique.

5.1 PERSPECTIVES

5.1.1 Parallélisation de COMPAREADS

Les résultats de l'outil COMPAREADS et son utilisation intensive dans le projet TARA OCEANS nous motivent à poursuivre son développement, particulièrement afin d'améliorer son temps d'exécution. COMPAREADS est une implémentation mono-cœur de notre méthodologie. Il n'est donc, pour le moment, pas parallélisé. Deux niveaux de parallélisation sont envisagés. Le premier, que l'on peut qualifier de "grain fin", concerne les fonctions de hachage utilisées dans le BDS. Actuellement, les quatre fonctions sont calculées séquentiellement. Notre structure autorise l'écriture simultanée des quatre bits d'information d'un k-mer sans risque d'accès concurrents. Pour un k-mer donné, on souhaite donc calculer les quatre fonctions de hachage et écrire les quatre bits correspondant dans le BDS de manière simultanée. Cette opération est réalisée plusieurs milliards de fois pour des gros jeux de données métagénomiques : le gain de temps sera sûrement significatif.

Le second niveau de parallélisation, qualifié de "gros grain", intervient lorsque l'on compare de gros jeux de données. Comme expliqué en amont (voir sous-section 3.3.1.4), avec les paramètres standards le BDS ne peut pas contenir l'ensemble des k-mers du jeu indexé : on doit alors découper l'index en plusieurs parties, traitées séquentiellement. La parallélisation gros grain qu'on souhaite mettre en place consiste à traiter simultanément plusieurs parties de l'index total du jeu. En terme de consommation mémoire, cette parallélisation est coûteuse. Actuellement, le BDS utilise 4 Go, avec les paramètres standards, pour indexer un milliard de k-mers. Pour indexer au complet un jeu provenant de TARA OCEANS (environ 7 milliards de k-mers), il faut 28 Go. Néanmoins, on peut alors réaliser l'ensemble des comparaisons entre un jeu et l'index complet en un temps comparable à réaliser seulement la comparaison du jeu avec la première partie de l'index. Cette optimisation sera surtout destinée à l'utilisation intensive de COMPAREADS sur des serveurs de calculs.

5.1.2 Redondance intra-échantillon

En comparant deux échantillons A et B à l'aide de COMPAREADS, il est courant que le pourcentage de séquences de $(A \overset{\rightsquigarrow}{\cap} B)$ soit différent du pourcentage de séquences de $(B \overset{\rightsquigarrow}{\cap} A)$. Ceci arrive car une unique séquence de A peut être similaire à plusieurs séquences de B, par exemple si ces séquences de B sont identiques entre elles. Notre méthode ne renseigne pas sur la composition d'un unique échantillon. Il peut être intéressant de disposer d'un score "d'autocorrélation" indiquant si certaines séquences d'un échantillon sont identiques ou similaires entre elles. Nous souhaiterions pouvoir appliquer COMPAREADS à un échantillon contre lui même. Évidement, avec notre méthode actuelle, un échantillon est similaire à lui même à 100%, car chaque séquence est similaire à au moins elle même dans le jeu de données.

SOUS-ÉCHANTILLONNAGE DU JEU Notre première idée pour analyser un jeu contre lui même est de procéder à un sous-échantillonnage aléatoire du jeu afin de le diviser en deux parties égales. On a ainsi 50% des séquences de l'échantillon dans un sous-échantillon a et les 50% restants dans un sous-échantillon b. En appliquant COMPAREADS sur a et b, on obtient un score d'autocorrélation, mais ce score n'est pas facilement interprétable.

Prenons le cas extrême où l'échantillon initial est composé pour moitié de séquences uniques et pour moitié d'une seule séquence, répétée un grand nombre de fois. En sous-échantillonnant aléatoirement cet échantillon, les sous-échantillons a et b sont composés pour moitié de sé-

quences uniques. Le score d'autocorrélation de COMPAREADS est alors de 50% pour $a \overset{\rightharpoonup}{\cap} b$ comme pour $b \overset{\rightharpoonup}{\cap} a$.

Prenons maintenant un cas extrême opposé : l'échantillon initial est composé pour moitié de séquences uniques et pour moitié de séquences en doublon, donc répétées deux fois. En sous-échantillonnant aléatoirement cet échantillon, chaque séquence a une probabilité $p = \frac{1}{2}$ d'être dans a et une probabilité $p = \frac{1}{2}$ d'être dans b. Ces probabilités sont indépendantes et deux séquences identiques ont alors une probabilité $p = \frac{1}{4}$ d'être ensemble dans a, une probabilité $p = \frac{1}{4}$ d'être ensemble dans b et une probabilité $p = \frac{1}{2}$ d'être séparées entre a et b. Statistiquement, a contient alors 50% de séquences uniques, 25% de séquences en doublon et 25% de séquences similaires à des séquences de b. Le score d'autocorrélation de COMPAREADS est alors de 25% pour $a \overset{\rightharpoonup}{\cap} b$ comme pour $b \overset{\rightharpoonup}{\cap} a$.

Dans les deux exemples précédents, le jeu de données est composé de 50% de séquences uniques et 50% de séquences répétées, mais le score d'autocorrélation est différent. Ce cas illustre la difficulté d'établir un score d'autocorrélation à l'aide d'un sous-échantillonnage, sans modifier COMPAREADS. Par contre, en prenant un score égale à $1 -$ autocorrélation, on obtient un indice de la quantité de séquences *différentes* dans l'échantillon, c'est-à-dire les séquences qu'il resterait en enlevant toutes les séquences répétées. Dans le premier exemple, il y a 50% de séquences uniques, plus une séquence répétée. On a donc 50% (et une) séquences différentes dans le jeu. Dans le second exemple, on a 50% de séquences uniques et 50% de séquences en doublon. Le score de 25% peut être vu comme indiquant que le jeu est composé à 75% de séquences différentes.

Le sous-échantillonnage 50%-50% n'est peut-être pas le plus adapté mais un sous-échantillonnage différent implique d'utiliser un nombre de séquences différent lors des phases d'indexation et de requête. La faisabilité et l'impact sur les faux positifs restent à définir.

MODIFICATION DU BDS Une autre possibilité est envisagée pour estimer le score d'autocorrélation. Le jeu au complet est indexé dans un BDS particulier. Chaque case de ce BDS particulier peut prendre la valeur 0, 1 ou 2, quand dans le BDS normal, les cases sont des bits et ne peuvent donc prendre que la valeur 0 ou 1. Lorsque l'on indexe un k-mer, on contrôle si les cases concernées dans le BDS sont toutes à 1 ou plus. Si elles sont toutes à 1 ou plus, ce k-mer a probablement déjà été identifié dans les séquences précédentes : on passe alors ces cases à 2. Si au moins une case a pour valeur 0, ce k-mer n'a pas déjà été identifié : on passe les cases contenant 0 à 1.

Une fois tous les k-mers indexés, chaque séquence est utilisée comme requête. Si une séquence a au moins t k-mers dont toutes les cases dans le BDS sont à 2, cette séquence est probablement similaire à une autre séquence du jeu, en plus d'elle même.

Un tel score d'autocorrélation n'est pas sensible au phénomène expliqué précédemment. Deux jeux composés de 50% de séquences uniques auront un score d'autocorrélation de 50%, que les séquences similaires soient répétées en doublon ou un très grand nombre de fois. De plus, les séquences similaires sont récupérées à la fin du processus : on peut alors décider d'enlever ces séquences pour obtenir un jeu composé seulement de séquences uniques selon les paramètres t et k. Mais, cette méthode présente deux inconvénients. Le premier est qu'un BDS permettant d'utiliser plus de deux valeurs (0 ou 1) consomme deux fois plus de mémoire que le BDS classique. Le second inconvénient est plus contraignant : les k-mers de toutes les séquences doivent être indexés simultanément dans le BDS. Or, la quantité de séquences qu'on peut indexer dans le BDS est limitée afin de contrôler le taux de faux positifs. Concrètement, en utilisant $k = 33$, on peut indexer 1 milliard de 33-mers, soit maximum 14,7 millions de séquences de 100 pb. Cette méthode ne peut donc pas être utilisée telle quelle sur des jeux de données de centaines de millions de séquences.

5.1.3 Sous-échantillonnage des jeux de données

Afin de réduire les temps de calcul lors de la comparaison de deux jeux de données A et B, nous avons envisagé de sous-échantillonner de manière aléatoire ces jeux. Deux approches sont envisageables.

SOUS-ÉCHANTILLONNAGE DE L'INDEX Prenons le cas ou l'on n'indexe que 50% du jeu B au lieu des 100% habituels dans COMPAREADS. La partie de B indexée est nommée B_1 et la partie non indexée, B_2. Si $A \overset{\rightsquigarrow}{\cap} B_1$ conduit à 10% de séquences similaires, statistiquement $A \overset{\rightsquigarrow}{\cap} B_2$ conduit aussi à 10%. Mais, on ne peut savoir si les séquences de A similaires à B_1 sont les mêmes, ou non, que les séquences de A similaires à B_2. Concernant $A \overset{\rightsquigarrow}{\cap} B$, on peut alors seulement conclure que le score de similarité a une borne inférieure : 10%. Cette borne n'est pas satisfaisante pour notre méthodologie, on ne peut donc pas sous-échantillonner le jeu à indexer.

SOUS-ÉCHANTILLONNAGE DU JEU REQUÊTE Sous-échantillonner le jeu requête est plus simple. On peut fixer un intervalle de confiance pour savoir combien de séquences du jeu requête doivent être tirées aléatoirement. Le sous-échantillonnage résultant estime alors correctement le score de similarité recherché. La première étape du pipeline complet de COMPAREADS se déroule de manière satisfaisante.

Mais, lors de la seconde étape, on indexe le résultat de $A \overset{\rightsquigarrow}{\cap} B$ (voir sous-section 3.2.2.3). Ces séquences sont issues d'un sous-échantillonnage et, comme montré précédemment, le sous-échantillonnage de l'index ne donne pas un résultat satisfaisant. Il n'est pas non plus possible d'indexer A en entier lors de la seconde étape, cela peut générer des faux positifs (voir sous-section 3.2.3.5).

Le sous-échantillonnage ne semble donc pas possible sans modifier notre méthode. De plus, ce sous-échantillonnage ne donnerait que le score de similarité entre deux échantillons et non les séquences similaires.

5.1.4 Un jeu contre N jeux

Les gros projets métagénomiques génèrent un grand nombre de jeux de données. Ces jeux ne sont pas tous générés au même moment, les technologies de séquençage nécessitant plusieurs heures à plusieurs jours pour traiter un échantillon. Ainsi, on peut souhaiter comparer un jeu de données A, fraichement séquencé, contre N jeux de données obtenus précédemment. Actuellement, COMPAREADS réaliserait toutes les comparaisons deux à deux. Il est possible d'optimiser ce traitement pour les deux premières étapes du pipeline complet de notre méthode (voir sous-section 3.2.2.3).

Lors de la première étape, les N jeux de données sont indexés séparément les uns des autres. On parcourt alors les séquences du jeu de données A une unique fois : chaque séquence est comparée contre les N index et on obtient N demi-intersections, une par index.

Lors de la seconde étape, on indexe A. Les N demi-intersections sont comparées contre cet index : on obtient N jeux résultats, contenant respectivement les séquences des N jeux de données similaires à au moins une séquence du jeu de données A.

Lors de la dernière étape, on indexe les N demi-intersections et chaque jeu résultat est comparé contre ces index. On obtient ainsi N autres jeux résultats : chacun de ces jeux résultats contient les séquences de A similaires à au moins une séquence de l'un des N jeux de données.

Ce processus permet, lors de la première étape, de ne lire qu'une seule fois le jeu de données A, au lieu de N fois en réalisant classiquement toutes les intersections deux à deux. De même, lors de la seconde passe, le jeu A n'est indexé qu'une seule fois, et non N fois. La dernière étape est par contre identique à faire toutes les comparaisons deux à deux. Cette méthode nécessite de disposer d'une grande quantité de mémoire, pour pouvoir indexer les N jeux de données simultanément.

5.1.5 N jeux contre N jeux

Le calcul de similarité deux à deux entre plusieurs échantillons est quadratique : pour x échantillons, il y a $\frac{x^2-x}{2}$ calculs à effectuer. Le projet Tara Oceans va créer environ 2000 échantillons, et avec un calcul d'une intersection entre deux échantillons d'une dizaine d'heures, il faudrait 19 990 000 heures de calcul, soit 2280,4 ans de calcul, en utilisant une seule machine. Deux méthodes sont actuellement développées pour réaliser directement toutes les comparaisons entre N jeux de données.

PREMIÈRE MÉTHODE La première méthode, développée par Guillaume Holley lors de son stage de fin de Master 2, est une approche statistique. Dans cette méthode, un grand nombre de k-mers est choisi pseudo-aléatoirement. À partir de ces k-mers, on crée une matrice contenant en ligne, chaque métagénome à comparer, et en colonne, chaque k-mer choisi. À l'intersection d'une ligne et d'une colonne, on trouve le comptage d'un k-mer dans un métagénome. Une analyse en composantes principales à base de " *kernels* ", dite aussi KPCA (K*ernel* P*rincipal* C*omponent* A*nalysis*) est menée sur cette matrice.

Cette KPCA permet d'extraire de la matrice des composantes principales, qui sont une combinaison non-linéaire des comptages de k-mers. Le nombre de composantes principales est très inférieur à celui des k-mers choisis : cela permet notamment de réduire la dimensionnalité des métagénomes. Alors, un clustering à base de recuit simulé, mené sur les composantes principales, regroupe les métagénomes les plus proches, c'est-à-dire les métagénomes partageant un grand nombre de k-mers ayant approximativement les mêmes comptages. Divers algorithmes et une méthode de *scoring* permettent de paramétrer cette méthode.

Les premiers résultats de cette méthode, nommée SMMACK pour S*tatistical* M*ethod for* M*etagenomes* A*nalysis by* C*ounting* K*-mers*, sont prometteurs. Ces résultats, entre autre sur les données de GOS, sont très proches de ceux obtenus à l'aide de COMPAREADS. En terme de performances, cette méthode a nécessité environ 10 Go de mémoire et 84 heures de calcul quand COMPAREADS n'utilise que 4 Go et 72 heures et 30 minutes de calcul. Néanmoins, sur 31 métagénomes de Tara Oceans, la méthode statistique a utilisé environ 10 Go et 72 heures de calcul (en utilisant 24 cœurs). COMPAREADS a nécessité 4 Go et plus de 6 jours de calcul (en utilisant 50 cœurs).

Ainsi, cette méthode statistique est bien plus efficace que COMPAREADS lorsque l'on souhaite comparer entre eux de nombreux et grands métagénomes. Le principal avantage de COMPAREADS est de fournir en sortie du processus les séquences similaires entre deux échantillons, permettant ainsi de lancer de nouvelles analyses sur les résultats d'intersections. La méthode statistique ne permet pas cela. On peut cependant envisager de coupler les deux méthodes : dans un premier temps, utiliser la méthode statistique pour réaliser toutes les comparaisons, puis dans un second temps lancer COMPAREADS seulement sur les échantillons pour lesquels on souhaite obtenir les séquences similaires.

La méthode SMMACK n'est pas encore finie, par exemple le choix des paramètres à utiliser lors des différentes phases n'est pas totalement arrêté. De plus, un effort doit être fait afin d'accélérer les traitements et de minimiser l'impact mémoire.

SECONDE MÉTHODE La seconde méthode est actuellement mise en œuvre par Élise Larsonneur au Genoscope (Évry). Dans ce projet, les N échantillons à comparer sont indexés simultanément.

L'intérêt principal de cette méthode est de n'indexer qu'une seule fois chaque jeu de données et non $N - 1$ fois comme le fait le pipeline normal de CompareADS. Par contre, cette méthode nécessite d'avoir une mémoire suffisamment importante pour indexer l'ensemble des intersections temporaires. Le Genoscope a en sa possession deux machines disposant chacune de 16 processeurs 10 cœurs et de 2 To mémoire. Chaque machine devrait pouvoir calculer toutes les intersections entre 23 jeux de données simultanément. En utilisant la puissance cumulée des deux machines, il sera sans doute possible de calculer simultanément les intersections de 32 jeux les uns contre les autres. Ce projet en est à ses débuts mais semble prometteur.

5.1.6 Le cœur des échantillons

La dernière perspective présentée dans ce manuscrit est liée aux intersections de plusieurs jeux entre eux. Plutôt que de réaliser toutes les intersections deux à deux entre N jeux, nous souhaiterions réaliser l'intersection totale de plusieurs jeux. Cette intersection totale reste encore à définir : cherche-t-on à obtenir les séquences similaires à tous les jeux, *i.e.* ayant au moins une occurrence par jeu, ou les séquences similaires à $N - 1$ jeux, à $N - 2$ jeux, etc ?

Cette intersection totale permet de récupérer les séquences présentes au cœur de tous les échantillons. D'un point de vue biologique, ces séquences peuvent correspondre à des organismes ubiquistes, c'est-à-dire présents dans tous les milieux étudiés, ou à des séquences identiques mais provenant d'organismes différents. Dans les deux cas, ces séquences sont intéressantes à étudier.

Comme notre méthode identifie les séquences similaires, ou identiques, entre les jeux de données, il est possible que deux séquences similaires puissent être assemblées ensemble. En ne travaillant que sur les séquences similaires à plusieurs jeux, on réduit la complexité des données à analyser, tout en augmentant la couverture des séquences restantes. Dès lors, il est peut-être possible d'assembler de manière satisfaisante ces séquences en utilisant les logiciels d'assemblage classiques. Les contigs obtenus pourraient correspondre à des gènes ou des organismes non-cultivables, mais présents dans tous les échantillons analysés. Il sera peut-être possible d'assembler ainsi, et de manière totalement *de novo*, des organismes non cultivés en laboratoire.

ANNEXE

En annexe se trouve la publication de COMPAREADS. Cet article a été publié dans BMC BIOINFORMATICS et est accessible à l'adresse suivante : http://www.biomedcentral.com/1471-2105/13/S19/S10

Le logiciel COMPAREADS est disponible à l'adresse suivante : http://colibread.inria.fr/software/compareads/

Compareads: comparing huge metagenomic experiments

Nicolas Maillet*[1], Claire Lemaitre[1], Rayan Chikhi[2], Dominique Lavenier[1], Pierre Peterlongo*[1]

[1]INRIA Rennes - Bretagne Atlantique / IRISA, EPI GenScale, Rennes, France
[2]ENS Cachan / IRISA, EPI GenScale, Rennes, France

Email: Nicolas Maillet*- nicolas.maillet@inria.fr; Claire Lemaitre - claire.lemaitre@inria.fr; Rayan Chikhi - rayan.chikhi@irisa.fr; Dominique Lavenier - dominique.lavenier@irisa.fr; Pierre Peterlongo*- pierre.peterlongo@inria.fr;

*Corresponding authors

Abstract

Background: Nowadays, metagenomic sample analyses are mainly achieved by comparing them with *a priori* knowledge stored in data banks. While powerful, such approaches do not allow to exploit unknown and/or "unculturable" species, for instance estimated at 99% for Bacteria.

Methods: This work introduces Compareads, a *de novo* comparative metagenomic approach that returns the reads that are similar between two possibly metagenomic datasets generated by High Throughput Sequencers. One originality of this work consists in its ability to deal with huge datasets. The second main contribution presented in this paper is the design of a probabilistic data structure based on Bloom filters enabling to index millions of reads with a limited memory footprint and a controlled error rate.

Results: We show that Compareads enables to retrieve biological information while being able to scale to huge datasets. Its time and memory features make Compareads usable on read sets each composed of more than 100 million Illumina reads in a few hours and consuming 4 GB of memory, and thus usable on today's personal computers.

Conclusion: Using a new data structure, Compareads is a practical solution for comparing *de novo* huge metagenomic samples. Compareads is released under the CeCILL license and can be freely downloaded from http://alcovna.genouest.org/compareads/.

Introduction

The past five years have seen the arrival of High Throughput Sequencing (HTS), also known as Next-Generation Sequencing (NGS). These technologies drastically lowered sequencing costs and increased sequencing throughput. They radically changed molecular biology and computational biology, as data generation is no longer a bottleneck. In fact, nowadays a major challenge is the analysis and interpretation of sequencing data [1]. HTS democratized access to sequencing to almost all biological labs over the world. It also opened the doors to new techniques such as ChipSeq [2], ClipSeq [3], RadSeq [4] and the topic of this work, metagenomics [5].

Metagenomics, also known as "environmental genomics", provides an alternative to traditional single-genome studies for exploring the microbial world. Most microorganisms (up to 99% of Bacteria [6]) are unknown and possibly "unculturable". Even if traditional genomics sequencing methods are well studied, they are not suited for environmental samples, because of the need to cultivate clones. By sequencing uncultured genomes directly from environmental samples, metagenomics offers new ways to study this unexplored diversity.

HTS technologies provide fragments of sequences (called reads) of length a few hundred base pairs without any information about the locus nor the orientation on the molecule they come from. In the metagenomic context, an additional difficulty comes from the fact that each read may belong to any species.

Nowadays, it is difficult to assemble complex metagenomes (such as soil or water metagenomes) into longer consensus sequences, because reads from different species may be merged into one chimeric sequence. Mende and colleagues [7] showed that for a 400-genomes metagenome, using simulated Illumina reads, 37% of the assembled sequences were chimeric. Thus currently, reads from metagenomes are used to estimate the biodiversity [8] or may be compared to known databases, providing information with respect to the current scientific knowledge [9,10]. Another way to exploit two or more metagenomic datasets is to compare them together, enabling to understand how genomic differences are related to environmental ones (biotopes localizations and/or time spent after an event).

Comparative metagenomics usually deals with many aspects, such as sequence composition, *i.e.* GC content [11], and genome size [12], taxonomic diversity [13], functional content [14], etc. Several methods are currently developed for comparative metagenomics analyses. Some are based on statistical methods with a large number of descriptive variables, *e.g.* principal component analysis (PCA).

To the best of our knowledge, there is no software designed to compare two or more metagenomic samples at the read level, *i.e.* to identify reads that are shared or similar between samples. This can be simply used

to compute a similarity measure between samples such as the number or percentage of similar reads between pairs of samples. When dealing with more than two samples, this would enable among others to classify metagenomics samples based on their raw reads content. One could use the popular tool BLAST to align reads in an all-vs-all way, however it is not designed specifically to this task, and more importantly, it cannot cope in time and memory with the size of nowadays metagenomic samples obtained with current sequencing technologies. For instance, with the aim of exploring the diversity of small eukaryotes in the oceans all over the world, the expedition "Tara Ocean" [15] is generating more than 400 metagenomic samples containing each around 100 million short reads, that will need to be compared to each other.

Here, we introduce a time and memory-efficient method for extracting similar reads between two metagenomic datasets. The similarity is based on shared k-mers (words of length k). In order to fit with current memory capacities, the data structure we use is a modified version of a Bloom filter [16]. Bloom filters have recently been used in bioinformatics, notably for assembly graph partitioning [17], which enabled to perform metagenomic *de novo* assembly using 30x less memory.

This manuscript presents two main contributions: **(I)** a new algorithm, called Compareads, which computes the similarity measure between two metagenomics datasets; **(II)** a new simple but extremely efficient data structure based on the Bloom filter for storing the presence/absence of k-mers in huge datasets. The manuscript is organized as follows: in Section 1, we depict the Compareads algorithm and the new data structure. In Section 2 we provide results both about the data structure and about Compareads, showing the efficiency of our approach in term of computation time, memory and biological accuracy.

1 Methods

Preliminaries and definitions A *sequence* is composed by zero or more symbols from an alphabet Σ. In this work, as we are dealing with DNA, $\Sigma = \{A, C, G, T\}$. A sequence s of length n on Σ is denoted also by $s[0]s[1]\dots s[n-1]$, where $s[i] \in \Sigma$ for $0 \leq i < n$. We denote by $s[i, j]$ the *substring* $s[i]s[i+1]\dots s[j]$ of s. In this case, we say that the substring $s[i, j]$ occurs at position i in s. We call *k-mer* a sequence of length k, and $s[i, i+k-1]$ is a k-mer occurring at position i in s.

Overview of Compareads Compareads is designed for finding similar sequences between two read sets. This basic operation may appear extremely simple. However, it has to be highly efficient, in term of computation time and memory footprint, in order to scale with huge metagenomics datasets.

In order to perform efficiently this operation, Compareads indexes k-mers and uses a rough but efficient notion of "*similar sequences*" defined as follows:

Definition 1 (shared k-mer) *Two sequences s_1 and s_2 share a k-mer if and only if $\exists(i_1, i_2)$ such that $s_1[i_1, i_1 + k - 1] = s_2[i_2, i_2 + k - 1]$.*

Definition 2 (Similar sequences) *Given integers k and t, two sequences s_1 and s_2 are said similar if and only if they share at least t non overlapping k-mers.*

In a few words, given two read sets A and B, the goal of the Compareads algorithm is to find the subset of reads from A which are similar to a read in B such set being denoted by $(A \overrightarrow{\cap} B)$. As it is a heuristic (see Section 1.4), our algorithm outputs an over-approximation of set $(A \overrightarrow{\cap} B)$ denoted by $(A \overrightarrow{\widetilde{\cap}} B)$.

1.1 Computing $(A \overrightarrow{\widetilde{\cap}} B)$

Compareads computes $(A \overrightarrow{\widetilde{\cap}} B)$ in two steps. The **indexing** step consists in storing in memory all k-mers having at least one occurrence in the set B. The **query** step processes reads from set A one by one. For a read $r \in A$, the index is used to test the presence in the set B of each k-mer of r. If at least t non-overlapping k-mers are returned as present, then the read r is inserted in $(A \overrightarrow{\widetilde{\cap}} B)$. The main practical challenge faced by Compareads is to index the possibly huge volume of k-mers contained in B. The data structure must therefore fulfill three criteria: it must be quick to build, have a low memory footprint and be quick to request. Section 1.2.2 describes the chosen probabilistic data structure, based on a Bloom filter.

Limiting the indexing space To control the approximation error (see Section 1.4), the indexing phase is interrupted whenever the volume of k-mers in the first reads of B exceeds a fixed value n. The query phase is then performed on the whole A dataset. This phase returns a partial intersection between A and a first chunk of reads from B. The remaining partial intersections between A and the next chunks of reads from B (each representing a volume of n k-mers or less) are sequentially computed, until all the reads from B have been indexed. Eventually, Compareads returns the union of all partial intersections. Note that, in terms of results, this partitioning approach is strictly equivalent to performing a complete indexing of B then a query of all the reads from A. To avoid redundant computations, reads from A considered as "similar" in one of the partial intersections are tagged using a bitvector and are not queried further.

Time complexity Let n_A and n_B be the number of k-mers respectively in set A and set B. Computing $(A \overrightarrow{\widetilde{\cap}} B)$ is done in time $O(n_B)$ (indexing) + $O\left(n_A \times \frac{n_B}{n}\right)$ (query). The $\frac{n_B}{n}$ term is due to the limitation of the indexing space.

1.2 Ad hoc data structure

The index data structure we use is based on a Bloom filter, specially designed for the task of storing efficiently a huge set of k-mers, while being fast to build and to query. We shortly recall in Section 1.2.1 what a Bloom filter is before describing, in Section 1.2.2, our data structure called BDS.

1.2.1 Bloom filter

A Bloom filter is a probabilistic data structure designed to test the membership of elements in a set [16]. It consists of an array of m bits, all initialized to zero, and a set of hash functions. Each hash function maps an element to a single position in the array. Each element is associated, through the values of the hash functions, to several positions in the array. To insert an element in the structure, the bits in the array associated to this element are all set to one. The structure answers membership queries by checking whether all the bits in the array associated to an element are set to one.

This data structure is probabilistic in nature, as false positives are possible. Even if an element is not in the set, its bits in the array may still be all set to one. This is because the bits associated to an element may independently be associated to other elements. Hence, the Bloom filter returns a wrong answer with non-zero probability. This probability is the *false positive rate*. An asymptotic approximation of the false positive rate is $0.6185^{m/n}$, assuming n elements are inserted in the m-bits array, and $(\ln 2 \cdot (m/n))$ hash functions are used [18]. False negatives never occur: if an element belongs to the set, the Bloom filter always answers positively. Bloom filters are space-efficient: only $(n \log_2 e \cdot \log_2(1/\epsilon))$ bits are required to support membership queries for n elements with a false positive rate of ϵ [18].

1.2.2 The Bloom Data Structure index

In this article, we consider a slightly different variation of Bloom filters: instead of using a single array of bits, each hash function corresponds to a distinct array, disjoint from all other functions. In terms of performance, with uniform hash functions, this variation is asymptotically equivalent to the original definition [18]. To avoid confusion with classical Bloom filters, we refer to this variation as BDS, standing for Bloom Data Structure.

Particular hash functions The hash functions used in this framework are a specific family of functions, which can be efficiently computed on consecutive k-mers. We consider the set of functions which map a k-mer to a bit sequence of length k, where each nucleotide is associated to a bit set to 0 or 1, depending only on its type (A, C, G or T). An exhaustive enumeration, in equations 1 and 2, shows that there exists only 7

functions in this set. We can distinguished two types, the first three, f_1, f_2 and f_3, are said to be *balanced* (equation 1), whereas the other four are said to be *unbalanced* (equation 2)

$$f_j : \Sigma^k \rightarrow \{0,1\}^k : \forall i \in [1,k] \begin{cases} f_1(s)[i] = 0 & \text{if} \quad s[i] = A \text{ or } C \quad f_1(s)[i] = 1 \quad \text{otherwise} \\ f_2(s)[i] = 0 & \text{if} \quad s[i] = A \text{ or } G \quad f_2(s)[i] = 1 \quad \text{otherwise} \\ f_3(s)[i] = 0 & \text{if} \quad s[i] = A \text{ or } T \quad f_3(s)[i] = 1 \quad \text{otherwise} \end{cases} \tag{1}$$

$$f_j : \Sigma^k \rightarrow \{0,1\}^k : \forall i \in [1,k] \begin{cases} f_4(s)[i] = 0 & \text{if} \quad s[i] = A \quad f_4(s)[i] = 1 \quad \text{otherwise} \\ f_5(s)[i] = 0 & \text{if} \quad s[i] = C \quad f_5(s)[i] = 1 \quad \text{otherwise} \\ f_6(s)[i] = 0 & \text{if} \quad s[i] = G \quad f_6(s)[i] = 1 \quad \text{otherwise} \\ f_7(s)[i] = 0 & \text{if} \quad s[i] = T \quad f_7(s)[i] = 1 \quad \text{otherwise} \end{cases} \tag{2}$$

One important property of these functions is that there is a simple relationship between the hash values of two consecutive k-mers in a read. One can see that the hash value of the next k-mer can be quickly computed, by left-shifting the binary sequence of the previous hash value and appending an extra bit. These functions are not classical hash functions, yet we show that they exhibit good hashing properties when applied to k-mers. In Section 2.1, the performance of these functions is compared with that of a classical hash function in terms of computation time, and false positive rate in the BDS.

1.3 The Compareads pipeline

Computing $(A \overrightarrow{\cap} B)$ is asymmetrical. Indeed $(A \overrightarrow{\cap} B)$ does not contain the reads from B which are similar to reads in A. For doing this, one needs to compute also $(B \overrightarrow{\cap} A)$. In practice, for fully and symmetrically comparing two sets A and B we apply a pipeline slightly more complicated than simply $(A \overrightarrow{\cap} B)$ followed by $(B \overrightarrow{\cap} A)$. This whole pipeline, designed for reducing a heuristic effect is described in Section 1.4.1.

Similarity measure While comparing read sets A and B, the result provided by Compareads is composed of two sets: $(A \overrightarrow{\cap} B)$ and $(B \overrightarrow{\cap} A)$. Then, a similarity measure between the two datasets is computed as follows: $Sim(A,B) = \frac{\left|A \overrightarrow{\cap} B\right| + \left|B \overrightarrow{\cap} A\right|}{|A|+|B|} * 100$, where $|X|$ denotes the cardinality of the set X.

1.4 Dealing with false positives

Our approach may generate false positives for two reasons we describe in the two upcoming sections, which also expose solutions for limiting these effects.

1.4.1 False positives due to k-mer shared between a read and a dataset

Using $t > 1$, Compareads algorithm can call similar sequences that do not respect strictly the definition of similarity given in definition 2. Indeed, steps described in Section 1.1 detect reads from A that share at least

t k-mers with reads from B. This is less stringent than finding reads from A that share at least t k-mers with at least one read from set B. In fact, the t k-mers found in read A are possibly spread over two or more distinct reads from set B.

This issue can be mitigated by performing the following steps to compute both $(A \overrightarrow{\cap} B)$ and $(B \overrightarrow{\cap} A)$:

1. Compute $(A \overrightarrow{\cap} B)$, storing the results in a set denoted by $(A \overrightarrow{\cap} B)^*$.
2. Compute $\left(B \overrightarrow{\cap} (A \overrightarrow{\cap} B)^*\right)$ storing the results in a set denoted by $(B \overrightarrow{\cap} A)$.
3. Compute $\left(A \overrightarrow{\cap} (B \overrightarrow{\cap} A)\right)$ storing the results in a set denoted by $(A \overrightarrow{\cap} B)$.

In a few words, the two output datasets $(B \overrightarrow{\cap} A)$ and $(A \overrightarrow{\cap} B)$ are obtained by applying the fundamental operation $(\overrightarrow{\cap})$ between a query and a read set being itself already the result of the asymmetrical $(\overrightarrow{\cap})$ operation. This enables to remove some false positives due to k-mers spread over several reads.

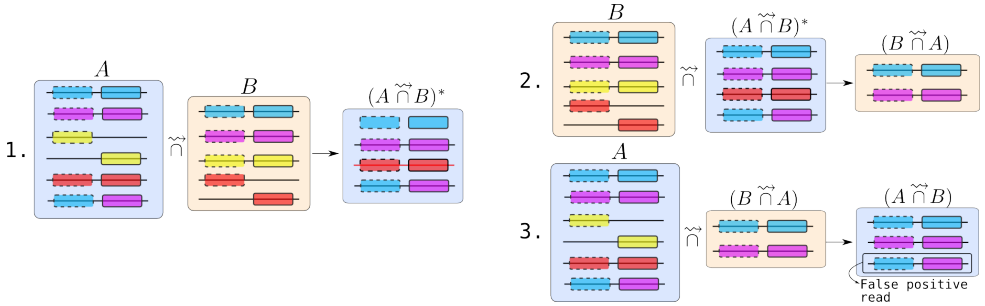


Figure 1: Representation of the three steps while comparing symmetrically read sets A and B. In each set, reads are represented by horizontal lines. On each read one or two shared k-mers are represented by rectangles.

The example presented in Figure 1 illustrates this issue for the case $t = 2$. The two first reads of sets A and B are similar. They are classically output by Compareads respectively in $(A \overrightarrow{\cap} B)$ and $(B \overrightarrow{\cap} A)$. The two next reads contain only one shared k-mer (yellow) with reads of set B, they are discarded. The next read of set A contains two (red) shared k-mers with two distinct reads in set B. After a first comparison, $(A \overrightarrow{\cap} B)^*$ contains this false positive read. However, in step 2, while computing $(B \overrightarrow{\cap} (A \overrightarrow{\cap} B)^*)$, these two reads are not conserved in $(B \overrightarrow{\cap} A)$. Thus, during step 3, the two red k-mers are not present anymore in set $(B \overrightarrow{\cap} A)$ and thus are not present in $(A \overrightarrow{\cap} (B \overrightarrow{\cap} A))$. They are thus correctly absent from the final results $(A \overrightarrow{\cap} B)$. However, the last read from set A is a case of false positive. It contains k-mers spread over distinct reads from B, the latter belonging to $(B \overrightarrow{\cap} A)$. Thus, even during step 3, these two k-mers remain shared with reads from set $(B \overrightarrow{\cap} A)$ and are output in $(A \overrightarrow{\cap} B)$.

Note that in practice, the last set $(A \overrightarrow{\cap} B)$ is obtained by computing $\left((A \overrightarrow{\cap} B)^* \overrightarrow{\cap} (B \overrightarrow{\cap} A)\right)$ instead of simply $\left(A \overrightarrow{\cap} (B \overrightarrow{\cap} A)\right)$ (used here for simplifying the reading). This operation provides the same result but is computed faster as $|(A \overrightarrow{\cap} B)^*| \leq |A|$.

As outlined in the example Figure 1, this pipeline still yields some false positives. These are characterized by t shared k-mers with at least two distinct reads from the indexed dataset B, themselves considered as similar to reads of set A. Even if this side effect is difficult to assess, we show in Section 2.2 that Compareads provides trustworthy results, highly similar to a classical approach, on several real datasets.

1.4.2 Bloom filter false positives

As exposed in Section 1.2.2, the BDS index is a probabilistic data structure, that may consider a k-mer as indexed while this is not the case (*i.e.* a false positive or FP). Here, we analyse the variations of the false positive rate for each hash function and their combinations with respect to the parameter k and the number n of distinct indexed k-mers. This enables to determine optimal parameters and appropriate combination of functions, that give the best trade-off between memory usage and false positive rate.

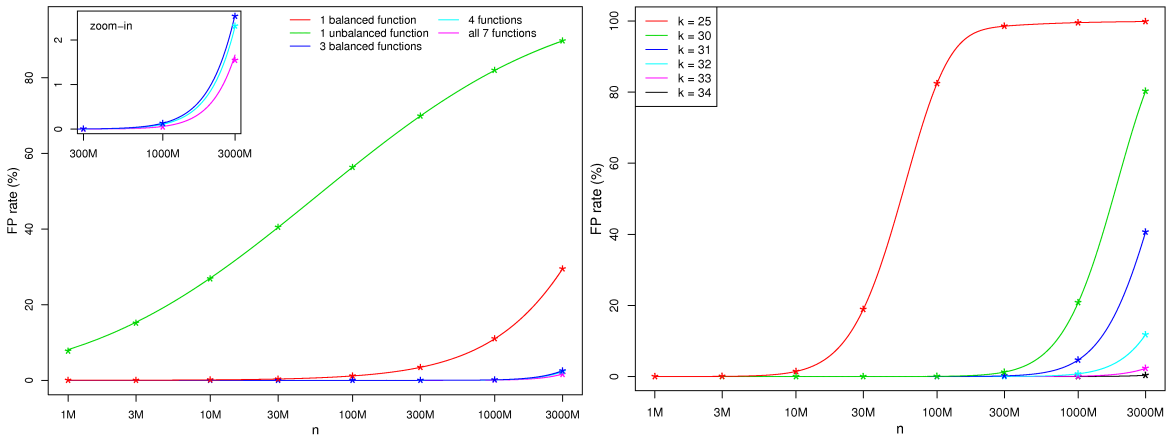
FP probablity for each function Assuming the nucleotide composition of the indexed k-mers and of the query k-mers are unbiaised, we can easily compute the probability, $P_{FP}(f_i, k, n)$, for any query k-mer to be a false positive with one of the seven hash functions, f_i (see the appendix Section A for details). The expression of this probablity is presented in equation 3 for a balanced hash function, and 4 for an unbalanced one.

$$\forall i \in \{1,2,3\} \quad P_{FP}(f_i, k, n) = 1 - (1 - \frac{1}{2^k})^n \tag{3}$$

$$\forall i \in \{4,5,6,7\} \quad P_{FP}(f_i, k, n) = \sum_{x=0}^{k} \binom{k}{x} a_x (1 - (1 - a_x)^n) \qquad with \quad a_x = (\frac{1}{4})^x (\frac{3}{4})^{k-x} \tag{4}$$

We have plotted in Figure 2a the theoretical FP rate for both types of hash functions, and we can see that balanced functions give much less false positives than unbalanced ones. This is due to the fact that balanced functions distribute the hash codes uniformly over the 2^k bit-array, while this not the case for the unbalanced ones.

FP probablity for a combination of functions One important property of the balanced hash functions is that there do not exist two distinct k-mers that have the same couple of hash codes with any two of these functions. This implies that, in terms of false positives, two balanced functions have limited interferences with each other (see details in appendix Section A). The probability of FP can then be easily computed as



(a) FP rate for different combinations of hash functions

(b) FP rate for different values of k

Figure 2: FP rate as a function of the number of indexed k-mers (in log scale). Plain lines correspond to theoretical predictions, whereas star points correspond to empirical values obtained with simulations. **(a)** This figure was obtained for $k = 33$ for balanced and unbalanced functions and some combinations of them. The combination entitled "4 functions" is composed of the 3 balanced functions plus one unbalanced. **(b)** For several values of k, this figure was obtained for a combination of 4 hash functions: all three balanced plus one unbalanced.

follows:

$$P_{FP}(f_1 \cap f_2 \cap f_3, k, n) \lesssim (1 - (1 - \frac{1}{2^k})^n)^3 \tag{5}$$

This "independence" property implies also that combining these 3 functions in our BDS is a very efficient strategy to reduce the FP rate, as can be seen in Figure 2a, especially for large values of n

Concerning the unbalanced functions, such property does not hold, since it is possible to find couples of distinct k-mers that share the same couple of hash codes for at least 2 of the unbalanced functions, or for one balanced function and at least one unbalanced. Therefore the theoretical FP rate of unbalanced functions is much more difficult to compute. We performed simulations to compute an empirical FP rate. We found that empirical results are very close to the formula obtained by multiplying the individual probabilities, *i.e.* assuming complete independence between all functions (Figure 2a). For details about how empirical results were obtained, see appendix Section B.

Choice of parameters The comparison of these FP curves led us to choose the combination of the three balanced functions plus an unbalanced one. This choice is motivated by the fact that unbalanced functions are not essential, as they have a limited effect on the FP rate (Figure 2a). Since hash functions described in Section 1.2.2 have a fixed range, the memory used by the BDS depends only on the value of k and the

number of hash functions used. Recall that each hash function is associated to a dedicated bit array which occupies 2^k bits. Using 7 hash functions, the BDS has a total memory footprint of $7 \cdot 2^k$ bits and can be stored in 2^k bytes. When using only 4 hash functions, the BDS occupies $4 \cdot 2^k$ bits and can be stored in twice less space (2^{k-1} bytes).

For the chosen combination of functions, we plotted the FP rate as a function of n and for several values of k in Figure 2b. The larger is k, the less FP we get for a given number of indexed k-mers. Consequently, for large values of k, more k-mers can be indexed while maintaining a reasonable FP rate. However, the memory allocated to BDS grows with k and larger values of k increases the stringency of our similarity measure. We can see in Figure 2b, that using k-mers of size at least 30 enables to index at least 300 millions of k-mers with less than 2% of false positives.

For $k = 33$, when indexing up to one billion distinct k-mers, we obtain a theoretical upper bound of 0.13% of false positives (with 3 balanced functions, equation 5). The FP rate is even lower when adding one of the unbalanced function, we estimated it empirically to 0.114%. Thus, using 4 hash functions and $k = 33$ is a good set of parameters for indexing one billion distinct k-mers. With such parameters, the memory usage of Compareads is 4 GB.

2 Results

2.1 Practical performance of the BDS, comparison with other data structures

We propose here a comparative analysis of the BDS with other data structures. In the following, we show that classical non probabilistic data structures result in a worse time and memory performance, while in Section 2.1.2, we show that the BDS is the best suited for the problem of indexing huge amounts of k-mers.

2.1.1 Comparison with non probabilistic data structures: suffix array and hash table

Indexing n characters using the simplest version of a suffix array (not enhanced [19] and without LCP information) requires $5n$ bytes of memory [20]. Compared to our set of parameters where $n = 1$ billion, the memory footprint would be 5×10^9 bytes, *i.e.* 4.66 GB. While this is comparable to the BDS, the query time of the suffix array, $O(k \log n)$, is significantly worse.

An hash table can be used to store an exact set of k-mers. Such structure stores the k-mers explicitly, hence it requires at least $n \cdot \lceil \frac{2k}{8} \rceil \cdot 8$ bits (assuming no overhead), i.e. 16.5 GB for one billion of 33-mers. Thus, the BDS is four times more succinct.

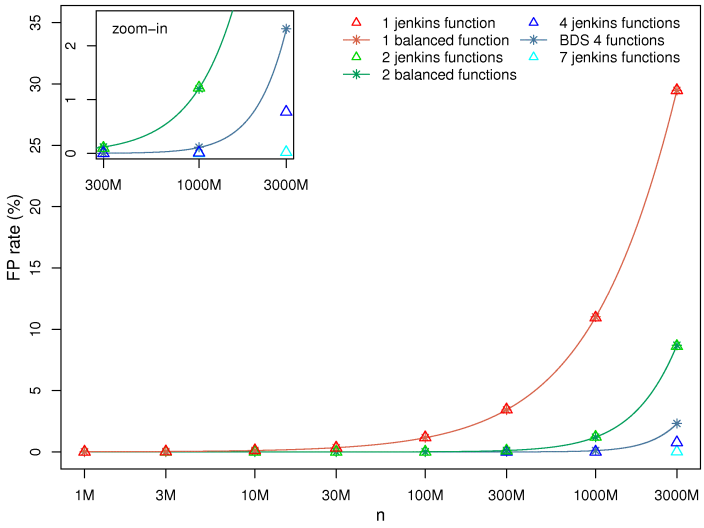


Figure 3: Comparison of FP rates between classical hash functions and the functions we used in the BDS. FP rate is plotted as a function of the number of indexed k-mers (in log scale), with $k = 33$. Plain lines correspond to theoretical predictions for the balanced functions (BDS), whereas star points and triangles correspond to empirical values obtained with simulations using respectively BDS functions and classical hash functions. The combination entitled "BDS 4 functions" is the one chosen for Compareads and is composed of the 3 balanced functions plus one unbalanced.

2.1.2 Comparison with other hash functions and with a classical Bloom filter

Time comparison with other hash functions The hash functions defined for BDS were designed with speed in mind. In this paragraph, we compare them with a popular and fast hash function (Jenkins hash, specifically `hashlittle2` from http://burtleburtle.net/bob/c/lookup3.c). We simulated 1 millon of 100-bp reads, where each nucleotide is drawn uniformly and independently. To simulate the behavior of computing hashes for the BDS, 4 hash values were computed for each 33-mer. For the `hashlittle2` function, we simulated this behavior by computing 4 hashes with 4 different initial values. We recorded the time required to compute the hashes for all the 33-mers present in the reads, averaged over 3 executions. Computing the hash with the `hashlittle2` function took 13.1 seconds (5.2 MHashes/s), whereas for the BDS hash functions, the same computation took 1.4 seconds (49.8 MHashes/s). Hence, the BDS hash functions are one order of magnitude faster than a classical set of hash functions.

FP rate comparison with other hash functions We can see in Figure 3 that the FP rate of classical hash functions follows the FP rate of our balanced functions (it follows the equation 5 with the exponent 3 being replaced by the number of functions used). However it diverges with more than 3 functions, as we could not add other balanced functions and we added in place unbalanced ones which have higher FP rates. Even if, for more than three functions, classical hash functions produce less FP, the difference with our BDS structure is small: for 1 billion indexed k-mers, combinations of 4 classical functions give 0.01% FP on average, compared to 0.114%. We chose to have a slithly higher FP rate, but with a significant gain in computing time.

Comparison with a classical Bloom filter A classical Bloom filter requires a fixed amount of memory to index n k-mers. Evaluating the Bloom filter memory using formula from Section 1.2.1 for one billion elements, with a false positive rate of 0.114%, yields 1.8 GB of memory. While this is twice smaller than the BDS, a classical Bloom filter would require classical hash functions. However, as shown above, classical hash functions are an order of magnitude slower to compute.

2.2 Comparing with a classical approach using Blast

Our approach is an heuristic based on shared k-mers between reads. Here we compare Compareads with a well-established method, Blast [21], that is based on sequence alignment. The dataset used is composed of 15 bacterial metagenomes obtained from fresh water with three different conditions of Carbon/Nitrogen ratio (unpublished data). On average, each sample is composed of 176409 reads with an average of 400 nucleotides per read (Roche 454 technology).

Both Blast and Compareads were used to compute all of the 120 pairwise intersections between the 15 datasets. Blast was configured to find similar sequences between two samples with a local alignment greater than 80 nucleotides and more than 90% of sequence identity. Compareads was used to find sequences sharing respectively t =1, 4 and 10 k-mers of 33 nucleotides. As shown in Table 1, computing one intersection

	Total Time (min)	Mean Time for one intersection (s)	Reads Found
Blast	7200	3600	33 400 091
Compareads $1 * 33$	238	119	35 898 023
Compareads $4 * 33$	230	115	31 997 243
Compareads $10 * 33$	228	114	21 350 268

Table 1: The CPU time per intersection and the global CPU time using a single core of an Intel® Xeon® CPU X5550 at 2.67GHz. **Reads Found** corresponds to the total number of similar reads in all the 120 intersections.

between two samples using Compareads is more than 30 times faster than using Blast for a close total number of similar reads.

For each experiment, samples were hierarchically clustered based on their pairwise similarity scores and then drawn as a dendrogram. As shown in Figure 4, the dendrogram obtained with the Blast approach (a) is slightly different but the three main branches are the same than with the Compareads approach (b). Interestingly, these branches discriminate three groups of samples corresponding to the three different biological conditions indicated by 1, 10 and 40 in the samples names: 1 corresponds to addition of Carbon in the water, 10 stands for normal condition and 40 for introduction of Nitrogen. Notably, all dendrograms

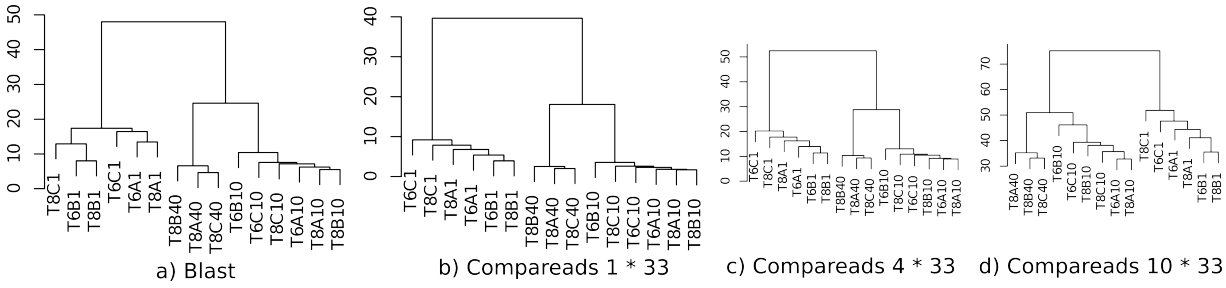


Figure 4: Representation of hierarchical clustering based on pairwise intersections between all samples using Blast (a) and Compareads (b, c, d).

based on Compareads approach (b, c, d) show a similar organization. Increasing the number of shared k-mers leads to be more stringent and decreases the number of similar reads but do not affect the global organization of the dendrogram, demonstrating the robustness of our similarity measure.

2.3 Applying Compareads to Global Ocean metagenomic samples

We tested Compareads on a larger and famous public dataset from the Global Ocean Sampling (The Sorcerer II expedition) [22]. It is composed of 44 samples from the microbial world of seawater, collected across several thousand of kilometers from the Northwest Atlantic through the Eastern Tropical Pacific oceans and for which an analysis of similarity between samples has been done [22]. The whole dataset is composed of 44 samples containing each on average 174759 long reads (1249 nucleotides per read on average, Sanger technology). Compareads computed all of the 990 intersections in 72 hours and half: on average, one intersection was performed in 4 minutes and 23 seconds on a single core of an Intel® Xeon® CPU X5550 at 2.67GHz. Results presented in Figure 5 are highly similar to those presented in the original publication [22], p.418. Two main groups are well discriminated. The first one, represented in turquoise-blue, groups together almost all samples coming from temperate seawater of the North American East Coast except the 14 one, consistently with the original study. This group also contains two samples really different from all others: the first contains freshwater and the second hypersaline water. The dark-green part corresponds to samples coming from the north part while light-green one gathers samples from the south part. Orange samples correspond to estuary. All of those three groups are identical to the original study. The second main part, colored in yellow, groups together datasets of tropical and Sargasso seawater. The dark-blue part aggregates samples coming only from Galapagos Islands. Red square delimitates Sargasso Sea samples. On the original study, the sample 00a is not in this group. According to metadata, the gray part, like in the original publication, is composed of various samples. Finally, purple samples regroup both Caribbean Sea and some

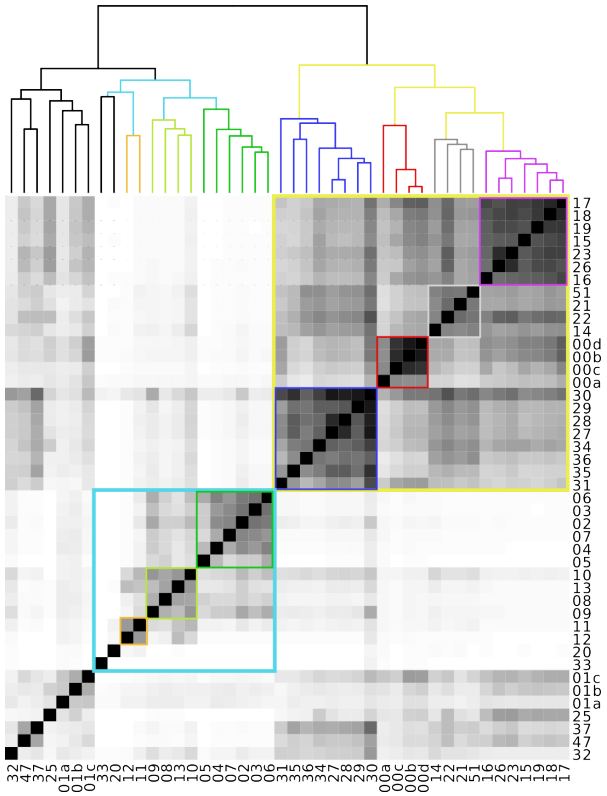


Figure 5: Similarity matrix resulting from the comparison of 44 samples from The *Sorcerer II* Global Ocean Sampling Expedition using Compareads. Grey levels correspond to similarity levels, intersections with more than 50% of similarity are in black. The two main groups, in turquoise-blue and yellow, correspond respectively to north American east coast and tropical samples.

Open Ocean datasets, as the original study.

Those results show that Compareads can also be used on Sanger reads and deliver reliable biological conclusions. Indeed, despite of false positives and the simple definition of similarity, we were able to retrieve the classification of metagenomes according to their geographical origin.

Conclusion

Motivated by *de novo* comparative metagenomics, this paper proposes two main contributions. The first one is a data structure based on Bloom filters that can index, for instance up to one billion distinct words of length 33 (33-mers) using 4Gb of memory, with an error rate of 0.11%, and that is faster to build and request, to the best of our knowledge, that any other existing data structure. The second main contribution is a software, called Compareads which uses this data structure to efficiently perform *de novo* intensive comparisons of huge metagenomic datasets generated by High Throughput Sequencers. We have shown that this approach enables to retrieve and classify differences in species content between metagenomic samples. For this kind of comparison, our approach is much faster than alternative ones such as BLAST and thus enables to scale to huge datasets. We furthermore tested the scalability of Compareads on a large oceanic

dataset (unpublished), from the Tara Ocean expedition [15]; it is composed of 31 metagenomes and contains overall 3.5 billions of Illumina short reads (108bp). Each intersection was performed in 10 hours and 55 minutes in average using 4Gb of memory. Such features enabled us to compute the $\frac{31*32}{2} = 496$ metagenome datasets intersections in 6 days and 10 hours using 50 cores of Intel® Xeon® CPU X5550 at 2.67GHz. This would have been unfeasible with any other known existing tools (based on results 2.2, BLAST is about 30 times longer and would take more than 6 months to complete this task with the same resources).

Compareads has been conceived for being parallelizable both at fine and coarse grained levels. Future work will consist in implementing a parallel version exploiting multi-core and GPU chips. Compareads is released under the CeCILL license and can be freely downloaded from http://alcovna.genouest.org/compareads/.

Competing interests

The authors declare that they have no competing interests.

Author's contributions

DL and PP initiated the work. RC and CL provided expertise about Bloom filters datastructures and their statistical aspects. NM and PP made the implementations. NM, CL, DL and PP performed the experiments. All authors participated to the redaction and approved the final manuscript.

Acknowledgements

This work was supported by the french ANR-2010-COSI-004 MAPPI Project. Authors warmly thank O. Jaillon from the Genoscope and P. Vandenkoornhuyse from the Ecobio UMR for providing their biological expertise and metagenomic datasets. Additionally, we thank G. Rizk for its help and comments with the data structure and F. Gauthier for the "$\widetilde{\cap}$" symbol creation.

References

1. Desai N, Antonopoulos D, Gilbert JA, Glass EM, Meyer F: **From genomics to metagenomics**. *Curr. Opin. Biotechnol.* 2012, **23**:72–76.
2. Johnson DS, Mortazavi A, Myers RM, Wold B: **Genome-wide mapping of in vivo protein-DNA interactions.** *Science (New York, N.Y.)* 2007, **316**(5830):1497–502.
3. Licatalosi DD, Mele A, Fak JJ, Ule J, Kayikci M, Chi SW, Clark TA, Schweitzer AC, Blume JE, Wang X, Darnell JC, Darnell RB: **HITS-CLIP yields genome-wide insights into brain alternative RNA processing**. *Nature* 2008, **456**(7221):464–469.
4. Davey JW, Blaxter ML: **RADSeq: next-generation population genetics**. *Briefings in Functional Genomics* 2010, **9**(5-6):416–423.

5. Wooley JC, Godzik A, Friedberg I: **A Primer on Metagenomics**. *PLoS Comput Biol* 2010, **6**(2):e1000667.

6. Amann RI, Ludwig W, Schleifer KH: **Phylogenetic identification and in situ detection of individual microbial cells without cultivation**. *Microbiol. Rev.* 1995, **59**:143–169.

7. Mende DR, Waller AS, Sunagawa S, Jäelin AI, Chan MM, Arumugam M, Raes J, Bork P: **Assessment of Metagenomic Assembly Using Simulated Next Generation Sequencing Data**. *PLoS ONE* 2012, **7**(2):e31386.

8. Wang Y, Leung HC, Yiu SM, Chin FY: **MetaCluster 4.0: a novel binning algorithm for NGS reads and huge number of species**. *J. Comput. Biol.* 2012, **19**(2):241–249.

9. Markowitz VM, Chen IM, Chu K, Szeto E, Palaniappan K, Grechkin Y, Ratner A, Jacob B, Pati A, Huntemann M, Liolios K, Pagani I, Anderson I, Mavromatis K, Ivanova NN, Kyrpides NC: **IMG/M: the integrated metagenome data management and comparative analysis system**. *Nucleic Acids Res.* 2011.

10. Huson DH, Auch AF, Qi J, Schuster SC: **MEGAN analysis of metagenomic data**. *Genome Res.* 2007, **17**(3):377–386.

11. Foerstner KU, von Mering C, Hooper SD, Bork P: **Environments shape the nucleotide composition of genomes**. *EMBO Rep.* 2005, **6**(12):1208–1213.

12. Raes J, Korbel JO, Lercher MJ, von Mering C, Bork P: **Prediction of effective genome size in metagenomic samples**. *Genome Biol.* 2007, **8**:R10.

13. Jaenicke S, Ander C, Bekel T, Bisdorf R, Droge M, Gartemann KH, Junemann S, Kaiser O, Krause L, Tille F, Zakrzewski M, Puhler A, Schluter A, Goesmann A: **Comparative and joint analysis of two metagenomic datasets from a biogas fermenter obtained by 454-pyrosequencing**. *PLoS ONE* 2011, **6**:e14519.

14. Sommer MO, Dantas G, Church GM: **Functional characterization of the antibiotic resistance reservoir in the human microflora**. *Science* 2009, **325**(5944):1128–1131.

15. Karsenti E: **Towards an 'Oceans Systems Biology'**. *Molecular Systems Biology* 2012, **8**(575):1–2.

16. Bloom BH: **Space/time trade-offs in hash coding with allowable errors**. *Commun. ACM* 1970, **13**(7):422–426.

17. Pell J, Hintze A, Brown T, Canino-koning R, Howe A, Tiedje JM: **Scaling metagenome sequence assembly with probabilistic de Bruijn graphs**. *PNAS*. in press.

18. Broder A, Mitzenmacher M: **Network applications of bloom filters: A survey**. *Internet Mathematics* 2004, **1**(4):485–509.

19. Abouelhoda MI, Kurtz S, Ohlebusch E: **Replacing suffix trees with enhanced suffix arrays**. *Journal of Discrete Algorithms* 2004, **2**:53–86.

20. Vyverman M, De Baets B, Fack V, Dawyndt P: **Prospects and limitations of full-text index structures in genome analysis.** *Nucleic acids research* 2012, :1–23.

21. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ: **Basic local alignment search tool**. *J. Mol. Biol.* 1990, **215**:403–410.

22. Rusch DB, Halpern AL, Sutton G, Heidelberg KB, Williamson S, Yooseph S, Wu D, Eisen JA, Hoffman JM, Remington K, Beeson K, Tran B, Smith H, Baden-Tillson H, Stewart C, Thorpe J, Freeman J, Andrews-Pfannkoch C, Venter JE, Li K, Kravitz S, Heidelberg JF, Utterback T, Rogers YH, Falcon LI, Souza V, Bonilla-Rosso G, Eguiarte LE, Karl DM, et al: **The Sorcerer II Global Ocean Sampling expedition: northwest Atlantic through eastern tropical Pacific**. *PLoS Biol.* 2007, **5**(3):e77.

Appendix

A Theoretical details for the false positive rate

As exposed in Section 1.2.2, the BDS index is a probabilistic data structure, that may consider a k-mer as indexed while this is not the case (*i.e.* a false positive). Here, we tried to express the false positive rate for each hash function that we defined in Section 1.2.2 and their combinations with respect to the parameter k and the number n of distinct indexed k-mers.

False positive probablity for each function Assuming the base composition of the indexed and query k-mers is unbiaised, we can easily compute the probability, $P_{FP}(f, k, n)$, for any query k-mer to be a false positive with one of the seven hash functions, f. This probability depends on the number of distinct k-mers sharing the same hash code. We can notice that for the balanced functions, f_1, f_2 and f_3, each 0 and 1 value can come from exactly 2 distinct nucleotides, thus the number of k-mers sharing the same hash code is the same for all k-mers and equals: 2^k. The probability for 2 k-mers to have distinct hash codes is then $1 - \frac{2^k}{4^k} = 1 - \frac{1}{2^k}$, and therefore the probability to have at least one k-mer among the n that are indexed sharing the same hash code is:

$$\forall i \in \{1, 2, 3\} P_{FP}(f_i, k, n) = 1 - (1 - \frac{1}{2^k})^n \qquad ((3))$$

Note that this corresponds to the false positive probability of any hash function that distributes the hash codes uniformly in a 2^k bit-array, such as those inspired of Jenkins functions, used as a comparison in Section 2.1.2.

As for the unbalanced functions, since the 0 bit-value encodes only one base, the number of k-mers sharing the same hash code depends on the number of 0 in the hash code of the query. For a given query k-mer with a hash code having x 0 the above probability for functions f_4, f_5, f_6 and f_7 becomes: $1-(1-(\frac{1}{4})^x(\frac{3}{4})^{(k-x)})^n$. To obtain the probability for any kmer, we have to sum over the different values of x the latter probability weigthed by the probablity for a k-mer hash code to have x 0. The composition of a given base in a k-mer of length k, assuming unbiased nucleotide composition, follows a binomial distribution, thus we get:

$$\forall i \in \{4, 5, 6, 7\} \quad P_{FP}(f_i, k, n) = \sum_{x=0}^{k} \binom{k}{x} a_x (1 - (1 - a_x)^n) \qquad with \quad a_x = (\frac{1}{4})^x (\frac{3}{4})^{k-x} \qquad ((4))$$

$\binom{k}{x}$ being the binomial coefficient, ie $\binom{k}{x} = \frac{k!}{x!(k-x)!}$.

We can see in Figure 2 that balanced functions give much less false positives than unbalanced ones. This can be explained by the fact that for unbalanced functions, for a given k-mer with a "normal" composition and thus 25% of 0 in its hash-code, there are many more k-mers with the same hash-code than for a balanced function: $3^{\frac{3k}{4}} \gg 2^k$.

FP probablity for a combination of functions When combining several functions in our BDS, in order to have a false positive for a query k-mer all functions must return a false positive. As concerns the balanced function, we can easily see that for a given kmer, we can not find another k-mer that is a false positive simultaneously for any two 2 of these functions. In other words, there do not exist two distinct k-mers that have the same couple of hash codes with any two of these functions. This implies that the probability of having a false positive with one function does not depend on the result with another function, apart from the fact that the effective number of indexed k-mers that can be a false positive (n in equation 3) is reduced: indeed if x k-mers have the same hash code for one function, these k-mers have a null probability of having the same hash code for another function. Note that this effect can be neglicted given that n is very large. Therefore the product of individual probabilities for each balanced function gives the following upper bound:

$$P_{FP}(f_1 \cap f_2 \cap f_3, k, n) \lesssim (1 - (1 - \frac{1}{2^k})^n)^3 \qquad ((5))$$

Concerning the unbalanced functions, this independence property is lost, since it is possible to find a single k-mer that is a false positive for at least 2 of the unbalanced functions, or for one balanced function and at least one unbalanced. Therefore we could not figure out the theoretical false positive rate, or even an upper bound.

B Empirical estimation of false positive rate

For estimating the false positive rate of the BDS filled up with n k-mers, we made the following experiment. We generated n+100000 distinct random k-mers, used first for filling up the BDS with n k-mers and then for querying the BDS with the 100000 remaining k-mers. All k-mers being distinct, if the BDS answers "yes" while querying the presence of a k-mer, it is a false positive.

The generation of a huge set of x distinct random k-mers (with $x < 4^k$) is not trivial. This was done by dividing the space of 4^k k-mers into x non overlapping blocks, and them by picking up a random k-mer into each block. This method enables to uniformly cover the whole k-mer space.

RÉFÉRENCES

[1] Narayan Desai, Dion Antonopoulos, Jack A Gilbert, Elizabeth M Glass, and Folker Meyer. From genomics to metagenomics. *Current Opinion in Biotechnology*, pages 1–5, Jan 2012.

[2] R Fitzroy and Sir J Barrow. *Sketch of the surveying voyages of His Majesty's Ships Adventure and Beagle, 1825-1836: commanded by Captain P.P. King, P. Stokes, and R. Fitz-Roy, Royal Navy / communicated by Sir John Barrow*. Journal of the Geological Society of London, 1836.

[3] Challenger (Ship), Sir J. Murray, G. S. Nares, Sir C. W. Thomson, and F. T. Thomson. *Report on the scientific results of the voyage of H.M.S. Challenger during the years 1873-76 : under the command of Captain George S. Nares, R.N., F.R.S. and Captain Frank Turle Thomson, R.N. / prepared under the superintendence of Sir C. Wyville Thomson.* Neill, Edinburgh, 1880.

[4] É. Karsenti and D. Di Meo. *Tara océans: Chroniques d'une expédition scientifique*. Actes Sud Editions, 2012.

[5] R.I Amann, W Ludwig, and K.H Schleifer. Phylogenetic identification and in situ detection of individual microbial cells without cultivation. *Microbiology and Molecular Biology Reviews*, 59(1):143, 1995.

[6] J. F. Miescher. Ueber die chemische Zusammensetzung der Eiterzellen. *Hoppe-Seyler's Medicinisch-chemische Untersuchungen*, 4:441–460, 1871.

[7] R. Altmann. Ueber Nucleinsaure. *Archiv fur Physiologie*, pages 524–536, 1889.

[8] A. Kossel. Ueber die basischen Stoffe des Zellkerns. *Hoppe-Seyler's Zeitschrift fur Physiologische Chemie*, 22:176–187, 1896.

[9] J. S. Cohen and Portugal H. The Search for the Chemical Structure of DNA. *Connecticut Medecine*, 38:551–557, 1974.

[10] P. Levene. The structure of yeast nucleic acid. *J. Biol. Chem.*, 40:415–424, dec 1919.

[11] W. Jones. *NUCLEIC ACIDS, Their Chemical Properties and Physiological Conduct*. Longmans, Green & co, New York, 1920.

[12] R. R. Feulgen. MMikroskopisch-chemischer Nachweis einer Nucleinsaure von Typus der Thymonucleinsaure und die darauf beruhende elektive Farbung von Zellkernen in mikroskopischer Praparaten. *Hoppe-Seyler's Zeitschrift fur Physiologische Chemie*, 135:203–248, 1924.

[13] W. Astbury. Nucleic acid. *Symp. SOC. Exp. Biol.*, 1, 1947.

[14] R. E. Franklin and Gosling R. Molecular Configuration in Sodium Thymonucleate. *Nature*, 171:740–741, 1953.

[15] R. E. Franklin and Gosling R. Evidence for 2-Chain Helix in Crystalline Structure of Sodium Deoxyribonucleate. *Nature*, 172:156–157, 1953.

[16] M. H. F. Wilkins, Stokes A., and Wilson H. Molecular Structure of Desoxypentose Nucleic Acids. *Nature*, 171:738–740, 1953.

[17] J. D. Watson and Crick F. H. C. Molecular structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid. *Nature*, 171:737–738, 1953.

[18] J. D. Watson and Crick F. H. C. Genetical implications of the structure of deoxyribonucleic acid. *Nature*, 171:964–967, 1953.

[19] J. Lejeune, M. Gauthier, and R. Turpin. Les chromosomes humains en culture de tissus. *Academie de Sciences*, 248, 1959.

[20] A M Maxam and W Gilbert. A new method for sequencing DNA. *Proceedings of the National Academy of Sciences*, 74(2):560–564, 1977.

[21] F. Sanger, S. Nicklen, and A. R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proc. Natl. Acad. Sci. U.S.A.*, 74(12):5463–5467, Dec 1977.

[22] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215(3):403–410, Oct 1990.

[23] A. Goffeau, B. G. Barrell, H. Bussey, R. W. Davis, B. Dujon, H. Feldmann, F. Galibert, J. D. Hoheisel, C. Jacq, M. Johnston, E. J. Louis, H. W. Mewes, Y. Murakami, P. Philippsen, H. Tettelin, and S. G. Oliver. Life with 6000 genes. *Science*, 274(5287):546–567, 1996.

[24] Frederick R. Blattner, Guy Plunkett, Craig A. Bloch, Nicole T. Perna, Valerie Burland, Monica Riley, Julio Collado-Vides, Jeremy D. Glasner, Christopher K. Rode, George F. Mayhew, Jason Gregor, Nelson Wayne Davis, Heather A. Kirkpatrick, Michael A. Goeden, Debra J. Rose, Bob Mau, and Ying Shao. The complete genome sequence of Escherichia coli K-12. *Science*, 277(5331):1453–1462, 1997.

[25] Genome Arabidopsis. Analysis of the genome sequence of the flowering plant Arabidopsis thaliana. *Nature*, 408(6814):796, 2000.

[26] Markus Covert, Eric Knight, Jennifer Reed, Markus Herrgard, and Bernhard Palsson. Integrating high-throughput and computational data elucidates bacterial networks. *Nature*, 429(6987):92–96, 2004.

[27] Wetterstrand KA. DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP), agu 2013.

[28] E. W Myers. A whole-genome assembly of Drosophila. *Science*, 287(5461):2196–2204, Mar 2000.

[29] Marcel Margulies, Michael Egholm, William E Altman, Said Attiya, Joel S Bader, Lisa A Bemben, Jan Berka, Michael S Braverman, Yi-Ju Chen, Zhoutao Chen, Scott B Dewell, Lei Du, Joseph M Fierro, Xavier V Gomes, Brian C Godwin, Wen He, Scott Helgesen, Chun He Ho, Gerard P Irzyk, Szilveszter C Jando, Maria L. I Alenquer, Thomas P Jarvie, Kshama B Jirage, Jong-Bum Kim, James R Knight, Janna R Lanza, John H Leamon, Steven M Lefkowitz, Ming Lei, Jing Li, Kenton L Lohman, Hong Lu, Vinod B Makhijani, Keith E Mcdade, Michael P Mckenna, Eugene W Myers, Elizabeth Nickerson, John R Nobile, Ramona Plant, Bernard P Puc, Michael T Ronan, George T Roth, Gary J Sarkis, Jan Fredrik Simons, John W Simpson, Maithreyan Srinivasan, Karrie R Tartaro, Alexander Tomasz, Kari A Vogt, Greg A Volkmer, Shally H Wang, Yong Wang, Michael P Weiner, Pengguang Yu, Richard F Begley, and Jonathan M Rothberg. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, pages 1–6, Jul 2005.

[30] Ruibang Luo, Binghang Liu, Yinlong Xie, Zhenyu Li, Weihua Huang, Jianying Yuan, Guangzhu He, Yanxiang Chen, Qi Pan, and Yunjie Liu. SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience*, 1(1):18, 2012.

[31] D. R. Zerbino and E. Birney. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 18(5):821, 2008.

[32] Rayan Chikhi and Guillaume Rizk. Space-efficient and exact de Bruijn graph representation based on a Bloom filter. In *WABI*, volume 7534 of *Lecture Notes in Computer Science*, pages 236–248. Springer, 2012.

[33] Jared Leadbetter. Cultivation of recalcitrant microbes: cells are alive, well and revealing their secrets in the 21st century laboratory. *Current opinion in microbiology*, 6(3):274–281, 2003.

[34] N. R. Pace, D. A. Stahl, and G. J. Olsen. Analyzing natural microbial populations by rRNA sequences. *ASM News*, 51:4–12, 1985.

[35] G J Olsen, D J Lane, S J Giovannoni, N R Pace, and D A Stahl. Microbial ecology and evolution: a ribosomal RNA approach. *Annu Rev Microbiol*, 40:337–65, Jan 1986.

[36] J Handelsman, M R Rondon, S F Brady, J Clardy, and R M Goodman. Molecular biological access to the chemistry of unknown soil microbes: a new frontier for natural products. *Chem Biol*, 5(10):R245–9, Oct 1998.

[37] JC Wooley, A Godzik, and I Friedberg. A primer on metagenomics. *PLoS Comput Biol*, 6(2):e1000667, 2010.

[38] Miguel Pignatelli and Andrés Moya. Evaluating the fidelity of de novo short read metagenomic assembly using simulated data. *PLoS ONE*, 6(5):e19984, May 2011.

[39] T Namiki, T Hachiya, H Tanaka, and Y Sakakibara. MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads. *Nucleic Acids Research*, 40(20):e155–e155, Nov 2012.

[40] M. Albertsen, P. Hugenholtz, A. Skarshewski, K. L. Nielsen, G. W. Tyson, and P. H. Nielsen. Genome sequences of rare, uncultured bacteria obtained by differential coverage binning of multiple metagenomes. *Nat. Biotechnol.*, 31(6):533–538, Jun 2013.

[41] T M Schmidt, E F DeLong, and N R Pace. Analysis of a marine picoplankton community by 16S rRNA gene cloning and sequencing. *J Bacteriol*, 173(14):4371–8, Jul 1991.

[42] Jonathan Kennedy, Burkhardt Flemer, Stephen A Jackson, David P. H Lejon, John P Morrissey, Fergal O'gara, and Alan D. W Dobson. Marine metagenomics: New tools for the study and exploitation of marine microbial metabolism. *Marine Drugs*, 8(3):608–628, Mar 2010.

[43] J Handelsman. Metagenomics: Application of genomics to uncultured microorganisms. *Microbiology and Molecular Biology Reviews*, 68(4):669–685, Dec 2004.

[44] Taku Uchiyama and Kentaro Miyazaki. Functional metagenomics for enzyme discovery: challenges to efficient screening. *Curr Opin Biotechnol*, 20(6):616–22, Dec 2009.

[45] Christopher Quince, Thomas P Curtis, and William T Sloan. The rational exploration of microbial diversity. *ISME J*, 2(10):997–1006, Oct 2008.

[46] Jiajia Ni, Qingyun Yan, and Yuhe Yu. How much metagenomic sequencing is enough to achieve a given goal? *Sci. Rep.*, 3:1–7, Jun 2013.

[47] S. Yilmaz, M. Allgaier, and P. Hugenholtz. Multiple displacement amplification compromises quantitative analysis of metagenomes. *Nat. Methods*, 7(12):943–944, Dec 2010.

[48] K. H. Kim and J. W. Bae. Amplification methods bias metagenomic libraries of uncultured single-stranded and double-stranded DNA viruses. *Appl. Environ. Microbiol.*, 77(21):7663–7668, Nov 2011.

[49] Sergei A Solonenko, J César Ignacio-Espinoza, Adriana Alberti, Corinne Cruaud, Steven Hallam, Kostas Konstantinidis, Gene Tyson, Patrick Wincker, and Matthew B Sullivan. Sequencing platform and library preparation choices impact viral metagenomes. *BMC Genomics*, 14(1):320, Jan 2013.

[50] H. Suenaga. Targeted metagenomics: a high-resolution metagenomics approach for specific gene clusters in complex microbial communities. *Environ. Microbiol.*, 14(1):13–22, Jan 2012.

[51] J. J. Grzymski, B. J. Carter, E. F. DeLong, R. A. Feldman, A. Ghadiri, and A. E. Murray. Comparative genomics of DNA fragments from six Antarctic marine planktonic bacteria. *Appl. Environ. Microbiol.*, 72(2):1532–1541, Feb 2006.

[52] D. Woebken, H. Teeling, P. Wecker, A. Dumitriu, I. Kostadinov, E. F. Delong, R. Amann, and F. O. Glockner. Fosmids of novel marine Planctomycetes from the Namibian and Oregon coast upwelling systems and their cross-comparison with Planctomycete genomes. *ISME J*, 1(5):419–435, Sep 2007.

[53] S. Demaneche, L. Philippot, M. M. David, E. Navarro, T. M. Vogel, and P. Simonet. Characterization of denitrification gene clusters of soil bacteria via a metagenomic approach. *Appl. Environ. Microbiol.*, 75(2):534–537, Jan 2009.

[54] K. A. Kazimierczak, K. P. Scott, D. Kelly, and R. I. Aminov. Tetracycline resistome of the organic pig gut. *Appl. Environ. Microbiol.*, 75(6):1717–1722, Mar 2009.

[55] Robert Schmieder and Robert Edwards. Insights into antibiotic resistance through metagenomic approaches. *Future Microbiol*, 7:73–89, Jan 2012.

[56] John Penders, Ellen E Stobberingh, Paul H. M Savelkoul, and Petra F. G Wolffs. The human microbiome as a reservoir of antimicrobial resistance. *Front. Microbiol.*, 4:1–7, Jan 2013.

[57] Alexandra M Schnoes, Shoshana D Brown, Igor Dodevski, and Patricia C Babbitt. Annotation error in public databases: Misannotation of molecular function in enzyme superfamilies. *PLoS Comput Biol*, 5(12):e1000605, Dec 2009.

[58] W. Ludwig, O. Strunk, R. Westram, L. Richter, H. Meier, Yadhukumar, , A. Buchner, T. Lai, S. Steppi, G. Jobb, W. Förster, I. Brettske, S. Gerber, AW. Ginhart, O. Gross, S. Grumann, S. Hermann, R. Jost, A. König, T. Liss, R. Lüssmann, M. May, B. Nonhoff, B. Reichel, R. Strehlow, A. Stamatakis, N. Stuckmann, A. Vilbig, M. Lenke, T. Ludwig, A. Bode, and KH. Schleifer. ARB: a software environment for sequence data. *Nucleic Acids Res.*, 32(4):1363–1371, 2004.

[59] R. C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, 32(5):1792–1797, 2004.

[60] J. R. Cole, Q. Wang, E. Cardenas, J. Fish, B. Chai, R. J. Farris, A. S. Kulam-Syed-Mohideen, D. M. McGarrell, T. Marsh, G. M. Garrity, and J. M. Tiedje. The Ribosomal Database Project: improved alignments and new tools for rRNA analysis. *Nucleic Acids Res.*, 37(Database issue):D141–145, Jan 2009.

[61] D Fimereli, V Detours, and T Konopka. TriageTools: tools for partitioning and prioritizing analysis of high-throughput sequencing data. *Nucleic Acids Research*, pages 1–8, Feb 2013.

[62] Dennis Benson, Ilene Karsch-Mizrachi, David Lipman, James Ostell, and David Wheeler. Genbank. *Nucleic Acids Research*, 33(suppl 1):D34–D38, 2005.

[63] Simon Roux, François Enault, Gisè Bronner, and Didier Debroas. Comparison of 16S rRNA and protein-coding genes as molecular markers for assessing microbial diversity (Bacteria and Archaea) in ecosystems. *FEMS Microbiol Ecol*, 78(3):617–628, Sep 2011.

[64] M. Kimura. *The Neutral Theory of Molecular Evolution*. Cambridge University Press, 1983.

[65] Eric Nawrocki, Diana Kolbe, and Sean Eddy. Infernal 1.0: inference of RNA alignments. *Bioinformatics*, 25(10):1335–1337, 2009.

[66] C. Manichanh, L. Rigottier-Gois, E. Bonnaud, K. Gloux, E. Pelletier, L. Frangeul, R. Nalin, C. Jarrin, P. Chardon, P. Marteau, J. Roca, and J. Dore. Reduced diversity of faecal microbiota in Crohn's disease revealed by a metagenomic approach. *Gut*, 55(2):205–211, Feb 2006.

[67] M. L Cuvelier, A. E Allen, A Monier, J. P Mccrow, M Messie, S. G Tringe, T Woyke, R. M Welsh, T Ishoey, J.-H Lee, B. J Binder, C. L Dupont, M Latasa, C Guigand, K. R Buck, J Hilton, M Thiagarajan, E Caler, B Read, R. S Lasken, F. P Chavez, and A. Z Worden. Targeted metagenomics and ecology of globally important uncultured eukaryotic phytoplankton. *Proceedings of the National Academy of Sciences*, 107(33):14679–14684, Aug 2010.

[68] M. S Lindner and B. Y Renard. Metagenomic abundance estimation and diagnostic testing on species level. *Nucleic Acids Research*, 41(1):e10–e10, Jan 2013.

[69] MetaHIT Consortium (additional Members. A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*, 464(7285):59–65, Apr 2010.

[70] J. A Frank and S. J Sorensen. Quantitative metagenomic analyses based on average genome size normalization. *Applied and Environmental Microbiology*, 77(7):2513–2521, Apr 2011.

[71] Kemal Sanli, Fredrik H Karlsson, Intawat Nookaew, and Jens Nielsen. FANTOM: Functional and taxonomic analysis of metagenomes. *BMC bioinformatics*, 14:38, Jan 2013.

[72] V. M Markowitz, N. N Ivanova, E Szeto, K Palaniappan, K Chu, D Dalevi, I-M. A Chen, Y Grechkin, I Dubchak, I Anderson, A Lykidis, K Mavromatis, P Hugenholtz, and N. C Kyrpides. IMG/M: a data management and analysis system for metagenomes. *Nucleic Acids Research*, 36(Database):D534–D538, Dec 2007.

[73] F Meyer, D Paarmann, M D'Souza, R Olson, Em Glass, M Kubal, T Paczian, A Rodriguez, R Stevens, A Wilke, J Wilkening, and Ra Edwards. The metagenomics RAST server – a public resource for the automatic phylogenetic and functional analysis of metagenomes. *BMC bioinformatics*, 9(1):386, Jan 2008.

[74] DH Huson, AF Auch, J Qi, and S. C Schuster. MEGAN analysis of metagenomic data. *Genome Research*, 17(3):377, 2007.

[75] M Arumugam, E. D Harrington, K. U Foerstner, J Raes, and P Bork. SmashCommunity: a metagenomic annotation and analysis tool. *Bioinformatics*, 26(23):2977–2978, Dec 2010.

[76] Q Wang, G. M Garrity, J. M Tiedje, and J. R Cole. Naive bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Applied and Environmental Microbiology*, 73(16):5261–5267, Aug 2007.

[77] G. Reinert, D. Chew, F. Sun, and M. S. Waterman. Alignment-free sequence comparison (I): statistics and power. *J. Comput. Biol.*, 16(12):1615–1634, Dec 2009.

[78] Min Li, Baohong Wang, Menghui Zhang, Mattias Rantalainen, Shengyue Wang, Haokui Zhou, Yan Zhang, Jian Shen, Xiaoyan Pang, Meiling Zhang, Hua Wei, Yu Chen, Haifeng Lu, Jian Zuo, Mingming Su, Yunping Qiu, Wei Jia, Chaoni Xiao, Leon M Smith, Shengli Yang, Elaine Holmes, Huiru Tang, Guoping Zhao, Jeremy K Nicholson, Lanjuan Li, and Liping Zhao. Symbiotic gut microbes modulate human metabolic phenotypes. *Proceedings of the National Academy of Sciences*, 105(6):2117–22, Feb 2008.

[79] Jonathan Kennedy, Julian R Marchesi, and Alan Dw Dobson. Marine metagenomics: strategies for the discovery of novel enzymes with biotechnological applications from marine environments. *Microb Cell Fact*, 7(1):27, Jan 2008.

[80] J Kennedy, N.D O'leary, G.S Kiran, J.P Morrissey, F O'gara, J Selvin, and A.D.W Dobson. Functional metagenomic strategies for the discovery of novel enzymes and biosurfactants with biotechnological applications from marine ecosystems. *Journal of Applied Microbiology*, 111(4):787–799, Aug 2011.

[81] M. O. A Sommer, G Dantas, and G. M Church. Functional characterization of the antibiotic resistance reservoir in the human microflora. *Science*, 325(5944):1128–1131, Aug 2009.

[82] R. Apweiler, M. Jesus Martin, C. O'onovan, M. Magrane, Y. Alam-Faruque, R. Antunes, E. Barrera Casanova, B. Bely, M. Bingley, L. Bower, B. Bursteinas, W. Mun Chan, G. Chavali, A. Da Silva, E. Dimmer, R. Eberhardt, F. Fazzini, A. Fedotov, J. Garavelli, L. G. Castro, M. Gardner, R. Hieta, R. Huntley, J. Jacobsen, D. Legge, W. Liu, J. Luo, S. Orchard, S. Patient, K. Pichler, D. Poggioli, N. Pontikos, S. Pundir, S. Rosanoff, T. Sawford, H. Sehra, E. Turner, T. Wardell, X. Watkins, M. Corbett, M. Donnelly, P. van Rensburg, M. Goujon, H. McWilliam, R. Lopez, I. Xenarios, L. Bougueleret, A. Bridge, S. Poux, N. Redaschi, G. Argoud-Puy, A. Auchincloss, K. Axelsen, D. Baratin, M. C. Blatter, B. Boeckmann, J. Bolleman, L. Bollondi, E. Boutet, S. Braconi Quintaje, L. Breuza, E. de-Castro, L. Cerutti, E. Coudert, B. Cuche, I. Cusin, M. Doche, D. Dornevil, S. Duvaud, A. Estreicher, L. Famiglietti, M. Feuermann, S. Gehant, S. Ferro, E. Gasteiger, V. Gerritsen, A. Gos, N. Gruaz-Gumowski, U. Hinz, C. Hulo, N. Hulo, J. James, S. Jimenez, F. Jungo, T. Kappler, G. Keller, V. Lara, P. Lemercier, D. Lieberherr, X. Martin, P. Masson, M. Moinat, A. Morgat, S. Paesano, I. Pedruzzi, S. Pilbout, M. Pozzato, M. Pruess, C. Rivoire, B. Roechert, M. Schneider, C. Sigrist, K. Sonesson, S. Staehli, E. Stanley, A. Stutz, S. Sundaram, M. Tognolli, L. Verbregue, A. L. Veuthey, C. H. Wu, C. N. Arighi, L. Arminski, W. C. Barker, C. Chen, Y. Chen, P. Dubey, H. Huang, A. Kukreja, K. Laiho, R. Mazumder, P. McGarvey, D. A. Natale, T. G. Natarajan, N. V. Roberts, B. E. Suzek, C. Vinayaka, Q. Wang, Y. Wang, L. S. Yeh, and J. Zhang. Reorganizing the protein space at the Universal Protein Resource (UniProt). *Nucleic Acids Res.*, 40(Database issue):D71–75, Jan 2012.

[83] M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. Hattori. The KEGG resource for deciphering the genome. *Nucleic Acids Res.*, 32(Database issue):D277–280, Jan 2004.

[84] M. Punta, P. C. Coggill, R. Y. Eberhardt, J. Mistry, J. Tate, C. Boursnell, N. Pang, K. Forslund, G. Ceric, J. Clements, A. Heger, L. Holm, E. L. Sonnhammer, S. R. Eddy, A. Bateman, and R. D. Finn. The Pfam protein families database. *Nucleic Acids Res.*, 40(Database issue):290–301, Jan 2012.

[85] J. D. Selengut, D. H. Haft, T. Davidsen, A. Ganapathy, M. Gwinn-Giglio, W. C. Nelson, A. R. Richter, and O. White. TIGRFAMs and Genome Properties: tools for the assignment of molecular function and biological process in prokaryotic genomes. *Nucleic Acids Res.*, 35(Database issue):D260–264, Jan 2007.

[86] S. Powell, D. Szklarczyk, K. Trachana, A. Roth, M. Kuhn, J. Muller, R. Arnold, T. Rattei, I. Letunic, T. Doerks, L. J. Jensen, C. von Mering, and P. Bork. eggNOG v3.0: orthologous groups covering 1133 organisms at 41 different taxonomic ranges. *Nucleic Acids Res.*, 40(Database issue):D284–289, Jan 2012.

[87] R. L. Tatusov, N. D. Fedorova, J. D. Jackson, A. R. Jacobs, B. Kiryutin, E. V. Koonin, D. M. Krylov, R. Mazumder, S. L. Mekhedov, A. N. Nikolskaya, B. S. Rao, S. Smirnov, A. V. Sverdlov, S. Vasudevan, Y. I. Wolf, J. J. Yin, and D. A. Natale. The COG database: an updated version includes eukaryotes. *BMC Bioinformatics*, 4:41, Sep 2003.

[88] Torsten Thomas, Jack Gilbert, and Folker Meyer. Metagenomics - a guide from sampling to data analysis. *Microbial Informatics and Experimentation*, 2(1):3, Feb 2012.

[89] Daniel H Huson and Chao Xie. A poor man's BLASTX - high-throughput metagenomic protein database search using PAUDA. *Bioinformatics (Oxford, England)*, May 2013.

[90] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. U.S.A.*, 89(22):10915–10919, Nov 1992.

[91] B. Langmead and S. L. Salzberg. Fast gapped-read alignment with Bowtie 2. *Nat. Methods*, 9(4):357–359, Apr 2012.

[92] S Ishii, M Yamamoto, M Kikuchi, K Oshima, M Hattori, S Otsuka, and K Senoo. Microbial populations responsive to denitrification-inducing conditions in rice paddy soil, as revealed by comparative 16S rRNA gene analysis. *Applied and Environmental Microbiology*, 75(22):7070–7078, Nov 2009.

[93] S Mirete, C. G De Figueras, and J. E Gonzalez-Pastor. Novel nickel resistance genes from the rhizosphere metagenome of plants adapted to acid mine drainage. *Applied and Environmental Microbiology*, 73(19):6001–6011, Oct 2007.

[94] Douglas B Rusch, Aaron L Halpern, Granger Sutton, Karla B Heidelberg, Shannon Williamson, Shibu Yooseph, Dongying Wu, Jonathan A Eisen, Jeff M Hoffman, Karin Remington, Karen Beeson, Bao Tran, Hamilton Smith, Holly Baden-Tillson, Clare Stewart, Joyce Thorpe, Jason Freeman, Cynthia Andrews-Pfannkoch, Joseph E Venter, Kelvin Li, Saul Kravitz, John F Heidelberg, Terry Utterback, Yu-Hui Rogers, Luisa I Falcón, Valeria Souza, Germán Bonilla-Rosso, Luis E Eguiarte, David M Karl, Shubha Sathyendranath, Trevor Platt, Eldredge Bermingham, Victor Gallardo, Giselle Tamayo-Castillo, Michael R Ferrari, Robert L Strausberg, Kenneth Nealson, Robert Friedman, Marvin Frazier, and J. Craig Venter. The sorcerer II Global Ocean Sampling Expedition: Northwest atlantic through eastern tropical pacific. *Plos Biol*, 5(3):e77, Jan 2007.

[95] Sebastian Jaenicke, Christina Ander, Thomas Bekel, Regina Bisdorf, Marcus Dröge, Karl-Heinz Gartemann, Sebastian Jünemann, Olaf Kaiser, Lutz Krause, Felix Tille, Martha Zakrzewski, Alfred Pühler, Andreas Schlüter, and Alexander Goesmann. Comparative and joint analysis of two metagenomic datasets from a biogas fermenter obtained by 454-pyrosequencing. *PLoS ONE*, 6(1):e14519, Jan 2011.

[96] Jesse A Port, James C Wallace, William C Griffith, and Elaine M Faustman. Metagenomic profiling of microbial composition and antibiotic resistance determinants in puget sound. *PLoS ONE*, 7(10):e48000, Oct 2012.

[97] Migun Shakya, Christopher Quince, James H Campbell, Zamin K Yang, Christopher W Schadt, and Mircea Podar. Comparative metagenomic and rRNA microbial diversity characterization using archaeal and bacterial synthetic communities. *Environmental Microbiology*, pages no–no, Feb 2013.

[98] Alexander M Cardoso, Janaína J. V Cavalcante, Maurício E Cantão, Claudia E Thompson, Roberto B Flatschart, Arnaldo Glogauer, Sandra M. N Scapin, Youssef B Sade, Paulo J. M. S. I Beltrão, Alexandra L Gerber, Orlando B Martins, Eloi S Garcia, Wanderley De Souza, and Ana Tereza R Vasconcelos. Metagenomic analysis of the microbiota from the crop of an invasive snail reveals a rich reservoir of novel genes. *PLoS ONE*, 7(11):e48505, Nov 2012.

[99] Konrad U Foerstner, Christian Von Mering, Sean D Hooper, and Peer Bork. Environments shape the nucleotide composition of genomes. *EMBO Rep*, 6(12):1208–1213, Dec 2005.

[100] Shibu Yooseph, Granger Sutton, Douglas B Rusch, Aaron L Halpern, Shannon J Williamson, Karin Remington, Jonathan A Eisen, Karla B Heidelberg, Gerard Manning, Weizhong Li, Lukasz Jaroszewski, Piotr Cieplak, Christopher S Miller, Huiying Li, Susan T Mashiyama, Marcin P Joachimiak, Christopher Van Belle, John-Marc Chandonia, David A Soergel, Yufeng Zhai, Kannan Natarajan, Shaun Lee, Benjamin J Raphael, Vineet Bafna, Robert Friedman, Steven E Brenner, Adam Godzik, David Eisenberg, Jack E Dixon, Susan S Taylor, Robert L Strausberg, Marvin Frazier, and J. Craig Venter. The sorcerer II Global Ocean Sampling Expedition: Expanding the universe of protein families. *Plos Biol*, 5(3):e16, Jan 2007.

[101] AT Sumner, Joaquina de la Torre, and L Stuppia. The distribution of genes on chromosomes: a cytological approach. *Journal of molecular evolution*, 37(2):117–122, 1993.

[102] Jeroen Raes, Jan O Korbel, Martin J Lercher, Christian Von Mering, and Peer Bork. Prediction of effective genome size in metagenomic samples. *Genome biology*, 8(1):R10, Jan 2007.

[103] W. J Kent. BLAT—the BLAST-Like alignment tool. *Genome Research*, 12(4):656–664, Mar 2002.

[104] R.C Edgar. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, 26(19):2460–2461, 2010.

[105] Bas E Dutilh, Robert Schmieder, Jim Nulton, Ben Felts, Peter Salamon, Robert A Edwards, and John L Mokili. Reference-independent comparative metagenomics using cross-assembly: crAss. *Bioinformatics*, Oct 2012.

[106] Jan Korbel, Berend Snel, Martijn Huynen, and Peer Bork. SHOT: a web server for the construction of genome phylogenies. *Trends in Genetics*, 18(3):158–162, 2002.

[107] William Wootters. Statistical distance and Hilbert space. *Physical Review D*, 23(2):357, 1981.

[108] S Dusko Ehrlich and MetaHIT Consortium. Metagenomics of the intestinal microbiota: potential applications. *Gastroentérologie Clinique et Biologique*, 34(S1):S23–S28, Sep 2010.

[109] Ruth Ley, Daniel Peterson, and Jeffrey Gordon. Ecological and evolutionary forces shaping microbial diversity in the human intestine. *Cell*, 124(4):837–848, 2006.

[110] Chaysavanh Manichanh, Lionel Rigottier-Gois, Elian Bonnaud, Karine Gloux, Eric Pelletier, Lionel Frangeul, Renaud Nalin, Cyrille Jarrin, Patrick Chardon, and Phillipe Marteau. Reduced diversity of faecal microbiota in Crohn's disease revealed by a metagenomic approach. *Gut*, 55(2):205–211, 2006.

[111] Peter J Turnbaugh, Ruth E Ley, Michael A Mahowald, Vincent Magrini, Elaine R Mardis, and Jeffrey I Gordon. An obesity-associated gut microbiome with increased capacity for energy harvest. *Nature*, 444(7122):1027–131, Dec 2006.

[112] J. G. Mulle, W. G. Sharp, and J. F. Cubells. The gut microbiome: a new frontier in autism research. *Curr Psychiatry Rep*, 15(2):337, Feb 2013.

[113] Manimozhiyan Arumugam, Jeroen Raes, Eric Pelletier, Denis Le Paslier, Takuji Yamada, Daniel R Mende, Gabriel R Fernandes, Julien Tap, Thomas Bruls, Jean-Michel Batto, Marcelo Bertalan, Natalia Borruel, Francesc Casellas, Leyden Fernandez, Laurent Gautier, Torben Hansen, Masahira Hattori, Tetsuya Hayashi, Michiel Kleerebezem, Ken Kurokawa, Marion Leclerc, Florence Levenez, Chaysavanh Manichanh, H. Bjørn Nielsen, Trine Nielsen, Nicolas Pons, Julie Poulain, Junjie Qin, Thomas Sicheritz-Ponten, Sebastian Tims, David Torrents, Edgardo Ugarte, Erwin G Zoetendal, Jun Wang, Francisco Guarner, Oluf Pedersen, Willem M De Vos, Søren Brunak, Joel Doré, María Antolín, François Artiguenave, Hervé M Blottiere, Mathieu Almeida, Christian Brechot, Carlos Cara, Christian Chervaux, Antonella Cultrone, Christine Delorme, Gérard Denariaz, Rozenn Dervyn, Konrad U Foerstner, Carsten Friss, Maarten Van De Guchte, Eric Guedon, Florence Haimet, Wolfgang Huber, Johan Van Hylckama-Vlieg, Alexandre Jamet, Catherine Juste, Ghalia Kaci, Jan Knol, Omar Lakhdari, Severine Layec, Karine Le Roux, Emmanuelle Maguin, Alexandre Mérieux, Raquel Melo Minardi, Christine M'rini, Jean Muller, Raish Oozeer, Julian Parkhill, Pierre Renault, Maria Rescigno, Nicolas Sanchez, Shinichi Sunagawa, Antonio Torrejon, Keith Turner, Gaetana Vandemeulebrouck, Encarna Varela, Yohanan Winogradsky, Georg Zeller, Jean Weissenbach, S. Dusko Ehrlich, and Peer Bork. Enterotypes of the human gut microbiome. *Nature*, 473(7346):174–180, May 2011.

[114] G. D Wu, J Chen, C Hoffmann, K Bittinger, Y.-Y Chen, S. A Keilbaugh, M Bewtra, D Knights, W. A Walters, R Knight, R Sinha, E Gilroy, K Gupta, R Baldassano, L Nessel, H Li, F. D Bushman, and J. D Lewis. Linking long-term dietary patterns with gut microbial enterotypes. *Science*, 334(6052):105–108, Oct 2011.

[115] Randy Ortíz-Castro, Hexon Contreras-Cornejo, Lourdes Macías-Rodríguez, and José López-Bucio. The role of microbial signals in plant growth and development. *Plant signaling & behavior*, 4(8):701–712, 2009.

[116] Tom O Delmont, Cedric Malandain, Emmanuel Prestat, Catherine Larose, Jean-Michel Monier, Pascal Simonet, and Timothy M Vogel. Metagenomic mining for microbiologists. pages 1–7, May 2011.

[117] Timothy Vogel, Pascal Simonet, Janet Jansson, Penny Hirsch, James Tiedje, Jan Van Elsas, Mark Bailey, Renaud Nalin, and Laurent Philippot. TerraGenome: a consortium for the sequencing of a soil metagenome. *Nature Reviews Microbiology*, 7(4):252, 2009.

[118] T. O Delmont, P Robe, S Cecillon, I. M Clark, F Constancias, P Simonet, P. R Hirsch, and T. M Vogel. Accessing the soil metagenome for studies of microbial diversity. *Applied and Environmental Microbiology*, 77(4):1315–1324, Feb 2011.

[119] Tom O Delmont, Patrick Robe, Ian Clark, Pascal Simonet, and Timothy M Vogel. Metagenomic comparison of direct and indirect soil DNA extraction approaches. *Journal of Microbiological Methods*, 86(3):397–400, Jul 2011.

[120] Tom O Delmont, Emmanuel Prestat, Kevin P Keegan, Michael Faubladier, Patrick Robe, Ian M Clark, Eric Pelletier, Penny R Hirsch, Folker Meyer, Jack A Gilbert, Denis Le Paslier, Pascal Simonet, and Timothy M Vogel. Structure, fluctuation and magnitude of a natural grassland soil metagenome. pages 1–11, Feb 2012.

[121] Tom O Delmont, Pascal Simonet, and Timothy M Vogel. Describing microbial communities and performing global comparisons in the 'omic era. 6(9):1625–1628, Jun 2012.

[122] Eric Karsenti, Silvia G Acinas, Peer Bork, Chris Bowler, Colomban De Vargas, Jeroen Raes, Matthew Sullivan, Detlev Arendt, Francesca Benzoni, Jean-Michel Claverie, Mick Follows, Gaby Gorsky, Pascal Hingamp, Daniele Iudicone, Olivier Jaillon, Stefanie Kandels-Lewis, Uros Krzic, Fabrice Not, Hiroyuki Ogata, Stéphane Pesant, Emmanuel Georges Reynaud, Christian Sardet, Michael E Sieracki, Sabrina Speich, Didier Velayoudon, Jean Weissenbach, and Patrick Wincker. A holistic approach to marine eco-systems biology. *Plos Biol*, 9(10):e1001177, Oct 2011.

[123] Eric Karsenti. Towards an 'oceans systems biology'. *Molecular Systems Biology*, 8(1), 2012.

[124] Edward McCreight. A space-economical suffix tree construction algorithm. *Journal of the ACM (JACM)*, 23(2):262–272, 1976.

[125] Udi Manber and Gene Myers. Suffix arrays: a new method for on-line string searches. pages 319–327, 1990.

[126] Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications. *Foundations of Computer Science*.

[127] B.H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[128] Esko Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

[129] M Vyverman, B De Baets, V Fack, and P Dawyndt. Prospects and limitations of full-text index structures in genome analysis. *Nucleic Acids Research*, 40(15):6993–7015, Aug 2012.

[130] Stefan Kurtz. Reducing the space requirement of suffix trees. *Software-Practice and Experience*, 29(13):1149–71, 1999.

[131] Mohamed Abouelhoda, Stefan Kurtz, and Enno Ohlebusch. Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2(1):53–86, 2004.

[132] Michael Burrows and David Wheeler. A block-sorting lossless data compression algorithm. *Digital SRC Research Report*, 1994.

[133] Nils Grimsmo. On performance and cache effects in substring indexes. 2007.

[134] J Pell, A Hintze, R Canino-Koning, A Howe, J.M Tiedje, and C.T Brown. Scaling metagenome sequence assembly with probabilistic de Bruijn graphs. *Arxiv preprint arXiv:1112.4193*, 2011.

[135] Andrei Broder and Michael Mitzenmacher. Network applications of Bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2004.

[136] Paul Jaccard. The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50, 1912.

[137] Y Fofanov, Y Luo, C Katili, J Wang, Y Belosludtsev, T Powdrill, C Belapurkar, V Fofanov, T.-B Li, S Chumakov, and B. M Pettitt. How independent are the appearances of n-mers in different genomes? *Bioinformatics*, 20(15):2421–2428, Oct 2004.

[138] Y Peng, H. C. M Leung, S. M Yiu, and F. Y. L Chin. Meta-IDBA: a de novo assembler for metagenomic data. *Bioinformatics*, 27(13):i94–i101, Jul 2011.

[139] G. Benson. Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Res.*, 27(2):573–580, Jan 1999.

[140] G. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, Jul 1948.